```
DDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBB      UUU         UUU     GGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG    GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG    GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG    GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG          GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG          GGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
```

FILEID**DBGNEXCTE

```
DDDDDDDD    BBBBBBBB      GGGGGGGG  NN      NN  EEEEEEEEEE  XX      XX    CCCCCCCC  TTTTTTTTTT  EEEEEEEEEE
DDDDDDDD    BBBBBBBB      GGGGGGGG  NN      NN  EEEEEEEEEE  XX      XX    CCCCCCCC  TTTTTTTTTT  EEEEEEEEEE
DD      DD  BB      BB  GG          NN      NN  EE          XX      XX  CC              TT      EE
DD      DD  BB      BB  GG          NN      NN  EE          XX      XX  CC              TT      EE
DD      DD  BB      BB  GG          NNNN    NN  EE            XX  XX    CC              TT      EE
DD      DD  BB      BB  GG          NNNN    NN  EE            XX  XX    CC              TT      EE
DD      DD  BBBBBBBB    GG          NN  NN  NN  EEEEEEEE        XX      CC              TT      EEEEEEEE
DD      DD  BBBBBBBB    GG          NN  NN  NN  EEEEEEEE        XX      CC              TT      EEEEEEEE
DD      DD  BB      BB  GG  GGGGGG  NN    NNNN  EE            XX  XX    CC              TT      EE
DD      DD  BB      BB  GG  GGGGGG  NN    NNNN  EE            XX  XX    CC              TT      EE
DD      DD  BB      BB  GG      GG  NN      NN  EE          XX      XX  CC              TT      EE                ....
DD      DD  BB      BB  GG      GG  NN      NN  EE          XX      XX  CC              TT      EE                ....
DDDDDDDD    BBBBBBBB      GGGGGG    NN      NN  EEEEEEEEEE  XX      XX    CCCCCCCC      TT      EEEEEEEEEE        ....
DDDDDDDD    BBBBBBBB      GGGGGG    NN      NN  EEEEEEEEEE  XX      XX    CCCCCCCC      TT      EEEEEEEEEE        ....
```

```
LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
;     1          0001  0 MODULE DBGNEXCTE (IDENT = 'V04-000') =
;     2          0002  0
;     3          0003  1 BEGIN
;     4          0004  1
;     5          0005  1 !
;     6          0006  1 !**********************************************************************
;     7          0007  1 !*                                                                    *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                           *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.            *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                             *
;    11          0011  1 !*                                                                    *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
;    17          0017  1 !*  TRANSFERRED.                                                       *
;    18          0018  1 !*                                                                    *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
;    21          0021  1 !*  CORPORATION.                                                       *
;    22          0022  1 !*                                                                    *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.            *
;    25          0025  1 !*                                                                    *
;    26          0026  1 !*                                                                    *
;    27          0027  1 !**********************************************************************
;    28          0028  1 !
;    29          0029  1 !
;    30          0030  1 ! FACILITY:     DEBUG
;    31          0031  1 !
;    32          0032  1 ! ABSTRACT:
;    33          0033  1 !
;    34          0034  1 !       Contained in this module is the routine DBG$NEXECUTE_CMD which uses the
;    35          0035  1 !       literal value ot the verb node of the command execution tree to decide
;    36          0036  1 !       which command execution network to invoke. In addition to this routine
;    37          0037  1 !       which is the highest level command execution network, this module contains
;    38          0038  1 !       several routines which are used by more than one command execution network
;    39          0039  1 !       during command execution.
;    40          0040  1 !
;    41          0041  1 ! ENVIRONMENT:  VAX/VMS
;    42          0042  1 !
;    43          0043  1 ! AUTHOR:       David Plummer, CREATION DATE:   4/15/80
;    44          0044  1 !
;    45          0045  1 ! VERSION:      V02.2-001
;    46          0046  1 !
;    47          0047  1 ! MODIFIED BY:
;    48          0048  1 !                Richard Title   Sep, 1981       Added support for the TYPE verb.
;    49          0049  1 !                RT              Oct, 1981       Added support for the SEARCH verb
;    50          0050  1 !                RT              Jan, 1982       Added support for the IF verb
;    51          0051  1 !                RT              Jan, 1982       Added support for the WHILE verb
;    52          0052  1 !                RT              Jan, 1982       Added support for the REPEAT verb
;    53          0053  1 !                RT              Jan, 1982       Added parameters to DBG$NCIS_ADD
;    54          0054  1 !                RT              Feb, 1982       Added support for EXITLOOP verb
;    55          0055  1 !                RT              Mar, 1982       Added support for DEFINE command
;    56          0056  1 !                RT              Apr, 1982       Added support for DECLARE command
;    57          0057  1 !                RT              Apr, 1982       Added support for SPAWN command
```

```
;    58          0058  1 !                    RT          May, 1982       Added support for ALLOCATE command
;    59          0059  1 !                    VJH         Jul, 1982       Added support for SYMBOLIZE command
;    60          0060  1 !                    RT          Aug, 1982       Changed DBG$NGET_ADDRESS to check
;    61          0061  1 !                                                for implementation level 3
;    62          0062  1 !                    RT          Sep, 1982       Added support for UNDEFINE command
;    63          0063  1 !                    PS          Oct, 1982       Added support for CALL command
;    64          0064  1 !                    RT          Dec, 1982       Added support for ATTACH command
;    65          0065  1 !                    RT          Feb, 1983       Added support for DUMP command
;    66          0066  1 !
;    67          0067  1
;    68          0068  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    69          0202  1
;    70          0203  1 LIBRARY 'LIB$:DBGGEN.L32';
;    71          0204  1
;    72          0205  1 FORWARD ROUTINE
;    73          0206  1     DBG$NEXECUTE_CMD,                   ! Highest level execution network
;    74          0207  1     DBG$NCIS_ADD,                       ! Adds a node to the CIS
;    75          0208  1     DBG$NCIS_OPENICF,                   ! Opens an icf node in the CIS
;    76          0209  1     DBG$NCIS_REMOVE,                    ! Removes a node from the CIS
;    77          0210  1     DBG$NGET_ADDRESS;                   ! Obtains an Lvalue or Rvalue
```

```
;    79          0211  1 EXTERNAL ROUTINE
;    80          0212  1     DBG$DEF_PR_EXIT,                ! Procedure exit for @ procedures
;    81          0213  1     DBG$DEF_SYM_ADD,                ! Add defined symbol
;    82          0214  1     DBG$DEF_SYM_FIND,               ! Look up defined symbol
;    83          0215  1     DBG$DEPOSIT: NOVALUE,           ! Level 3 EXECUTE_DEPOSIT routine
;    84          0216  1     DBG$EVALUATE: NOVALUE,          ! Level 3 EXECUTE_EVALUATE routine
;    85          0217  1     DBG$EXAMINE : NOVALUE,          ! Level 3 EXECUTE_EXAMINE routine
;    86          0218  1     DBG$GET_MEMORY,                 ! Allocate permanent memory
;    87          0219  1     DBG$GET_TEMPMEM,                ! Allocate temporary memory
;    88          0220  1     DBG$MAKE_VMS_DESC,              ! Convert Primary Descriptor to
;    89          0221  1                                     !    VMS Descriptor
;    90          0222  1     DBG$NCOPY_DESC,                 ! Copy a descriptor
;    91          0223  1     DBG$NEXECUTE_ALLOCATE,          ! ALLOCATE command execution network
;    92          0224  1     DBG$NEXECUTE_AT_SIGN,           ! @ filespec execution network
;    93          0225  1     DBG$NEXECUTE_ATTACH,            ! ATTACH command execution network
;    94          0226  1     DBG$NEXECUTE_CALL,              ! CALL command execution network
;    95          0227  1     DBG$NEXECUTE_CANCEL,            ! CANCEL command execution network
;    96          0228  1     DBG$NEXECUTE_DECLARE,           ! DECLARE command execution network
;    97          0229  1     DBG$NEXECUTE_DEFINE,            ! DEFINE command execution network
;    98          0230  1     DBG$NEXECUTE_DELETE,            ! DELETE command execution network
;    99          0231  1     DBG$NEXECUTE_DUMP,              ! DUMP command execution network
;   100          0232  1     DBG$NEXECUTE_EDIT,              ! EDIT command execution network
;   101          0233  1     DBG$NEXECUTE_EXIT,              ! EXIT command execution network
;   102          0234  1     DBG$NEXECUTE_EXITLOOP,          ! EXITLOOP command execution network
;   103          0235  1     DBG$NEXECUTE_FOR,               ! FOR command execution network
;   104          0236  1     DBG$NEXECUTE_GO,                ! GO command execution network
;   105          0237  1     DBG$NEXECUTE_HELP,              ! HELP command execution network
;   106          0238  1     DBG$NEXECUTE_IF,                ! IF command execution network
;   107          0239  1     DBG$NEXECUTE_REPEAT,            ! REPEAT command execution network
;   108          0240  1     DBG$NEXECUTE_SEARCH,            ! SEARCH command execution network
;   109          0241  1     DBG$NEXECUTE_SET,               ! SET verb execution network
;   110          0242  1     DBG$NEXECUTE_SHOW,              ! SHOW verb execution network
;   111          0243  1     DBG$NEXECUTE_SPAWN,             ! SPAWN verb execution network
;   112          0244  1     DBG$NEXECUTE_STEP,              ! STEP command execution network
;   113          0245  1     DBG$NEXECUTE_SYMBOLIZE,         ! SYMBOLIZE command execution network
;   114          0246  1     DBG$NEXECUTE_TYPE,              ! TYPE command execution network
;   115          0247  1     DBG$NEXECUTE_UNDEFINE,          ! UNDEFINE command execution network
;   116          0248  1     DBG$NEXECUTE_WHILE,             ! WHILE command execution network
;   117          0249  1     DBG$NFREE_DESC,                 ! Release space for a descriptor
;   118          0250  1     DBG$NGET_LVAL,                  ! Obtains a symbol's lvalue from a prim desc
;   119          0251  1     DBG$NGET_SYMID,                 ! Obtain a symid list
;   120          0252  1     DBG$NGET_TYPE,                  ! Obtains a symbol's type form a prim desc
;   121          0253  1     DBG$NMAKE_ARG_VECT,             ! Constructs a message argument vector
;   122          0254  1     DBG$NOUT_INFO,                  ! Outputs an info message
;   123          0255  1     DBG$REL_MEMORY: NOVALUE,        ! Release permanent memory
;   124          0256  1     DBG$SCR_EXECUTE_DISPLAY_CMD:NOVALUE,! Execute the DISPLAY command
;   125          0257  1     DBG$SCR_EXECUTE_SAVE_CMD: NOVALUE,  ! Execute the SAVE command
;   126          0258  1     DBG$SCR_EXECUTE_SCROLL_CMD: NOVALUE,! Execute the SCROLL command
;   127          0259  1     DBG$SCR_EXECUTE_SELECT_CMD: NOVALUE,! Execute the SELECT command
;   128          0260  1     DBG$STA_LOCK_SYMID: NOVALUE;    ! Lock a SYMID list
;   129          0261  1
;   130          0262  1 EXTERNAL
;   131          0263  1     DBG$GL_CISHEAD: REF CIS$LINK,   ! Version 2 debugger head of command input stream
;   132          0264  1     DBG$GL_CIS_LEVELS,              ! Count of number of levels of CIS.
;   133          0265  1     DBG$GB_DEF_OUT: VECTOR[,BYTE],  ! Old debugger output vector control
;   134          0266  1     DBG$GL_SCREEN_ERROR,            ! Screen error display pointer (or 0)
;   135          0267  1     DBG$GL_SCREEN_NOGO,             ! Screen flag to turn off STEP and GO
```

```
;  136          0268  1      DBG$GL_SCREEN_OUTPUT,                   ! Screen output display pointer (or 0)
;  137          0269  1      DBG$GL_SCREEN_SOURCE;                   ! Screen source display pointer (or 0)
;  138          0270  1
;  139          0271  1  LITERAL
;  140          0272  1      ALLOCATE_VERB          = DBG$K_ALLOCATE_VERB,
;  141          0273  1      AT_SIGN_VERB           = DBG$K_AT_SIGN_VERB,
;  142          0274  1      ATTACH_VERB            = DBG$K_ATTACH_VERB,
;  143          0275  1      CALL_VERB              = DBG$K_CALL_VERB,
;  144          0276  1      CANCEL_VERB            = DBG$K_CANCEL_VERB,
;  145          0277  1      DECLARE_VERB           = DBG$K_DECLARE_VERB,
;  146          0278  1      DEFINE_VERB            = DBG$K_DEFINE_VERB,
;  147          0279  1      DELETE_VERB            = DBG$K_DELETE_VERB,
;  148          0280  1      DEPOSIT_VERB           = DBG$K_DEPOSIT_VERB,
;  149          0281  1      DISPLAY_VERB           = DBG$K_DISPLAY_VERB,
;  150          0282  1      DUMP_VERB              = DBG$K_DUMP_VERB,
;  151          0283  1      EDIT_VERB              = DBG$K_EDIT_VERB,
;  152          0284  1      EVALUATE_VERB          = DBG$K_EVALUATE_VERB,
;  153          0285  1      EXAMINE_VERB           = DBG$K_EXAMINE_VERB,
;  154          0286  1      EXIT_VERB              = DBG$K_EXIT_VERB,
;  155          0287  1      EXITLOOP_VERB          = DBG$K_EXITLOOP_VERB,
;  156          0288  1      FOR_VERB               = DBG$K_FOR_VERB,
;  157          0289  1      GO_VERB                = DBG$K_GO_VERB,
;  158          0290  1      HELP_VERB              = DBG$K_HELP_VERB,
;  159          0291  1      IF_VERB                = DBG$K_IF_VERB,
;  160          0292  1      REPEAT_VERB            = DBG$K_REPEAT_VERB,
;  161          0293  1      SAVE_VERB              = DBG$K_SAVE_VERB,
;  162          0294  1      SCROLL_VERB            = DBG$K_SCROLL_VERB,
;  163          0295  1      SEARCH_VERB            = DBG$K_SEARCH_VERB,
;  164          0296  1      SELECT_VERB            = DBG$K_SELECT_VERB,
;  165          0297  1      SET_VERB               = DBG$K_SET_VERB,
;  166          0298  1      SHOW_VERB              = DBG$K_SHOW_VERB,
;  167          0299  1      SPAWN_VERB             = DBG$K_SPAWN_VERB,
;  168          0300  1      STEP_VERB              = DBG$K_STEP_VERB,
;  169          0301  1      SYMBOLIZE_VERB         = DBG$K_SYMBOLIZE_VERB,
;  170          0302  1      TYPE_VERB              = DBG$K_TYPE_VERB,
;  171          0303  1      UNDEFINE_VERB          = DBG$K_UNDEFINE_VERB,
;  172          0304  1      WHILE_VERB             = DBG$K_WHILE_VERB;
;  173          0305  1
;  174          0306  1
;  175          0307  1  ! The following macro verifies entrance to, or exit from an ICF.
;  176          0308  1  !
;  177          0309  1  MACRO
;  178        M 0310  1          ICF_MESSAGE (PREFIX) =
;  179        M 0311  1
;  180        M 0312  1          BEGIN
;  181        M 0313  1          BIND
;  182        M 0314  1                  ENTER_PHRASE = UPLIT BYTE(8, %ASCII 'entering'),
;  183        M 0315  1                  EXIT_PHRASE  = UPLIT BYTE(7, %ASCII 'exiting');
;  184        M 0316  1
;  185        M 0317  1          LOCAL
;  186        M 0318  1                  PHRASE;
;  187        M 0319  1
;  188        M 0320  1          IF prefix EQL 1
;  189        M 0321  1          THEN
;  190        M 0322  1                  phrase = enter_phrase
;  191        M 0323  1          ELSE
;  192        M 0324  1                  phrase = exit_phrase;
```

```
;  193          M 0325  1
;  194          M 0326  1
;  195          M 0327  1      dbg$nout_info (dbg$_verifyicf, 3, .phrase, .fab_ptr[fab$b_fns], .fab_ptr[fab$l_fna]);    ! Info messa
;  196          M 0328  1
;  197            0329  1      END %;
```

```
;  199     0330  1 GLOBAL ROUTINE DBG$NEXECUTE_CMD (VERB_NODE_PTR, MESSAGE_VECT) =
;  200     0331  1
;  201     0332  1 !
;  202     0333  1 ! FUNCTIONAL DESCRIPTION:
;  203     0334  1 !       DBG$NEXECUTE_CMD is the highest level command execution network. This
;  204     0335  1 !       routine examines the value of the verb node in the command execution
;  205     0336  1 !       tree to decide which DEBUG command is to be executed, and transfer to
;  206     0337  1 !       an appropriate subnetwork to perform the associated semantic action.
;  207     0338  1 !
;  208     0339  1 ! FORMAL PARAMETERS:
;  209     0340  1 !
;  210     0341  1 !       VERB_NODE_PTR -         pointer to the head of the command execution tree
;  211     0342  1 !
;  212     0343  1 !       MESSAGE_VECT -          address of a longword to contain the address of
;  213     0344  1 !                               a message argument vector
;  214     0345  1 !
;  215     0346  1 ! IMPLICIT INPUTS:
;  216     0347  1 !
;  217     0348  1 !       NONE
;  218     0349  1 !
;  219     0350  1 ! IMPLICIT OUTPUTS:
;  220     0351  1 !
;  221     0352  1 !       NONE
;  222     0353  1 !
;  223     0354  1 ! ROUTINE VALUE:
;  224     0355  1 !
;  225     0356  1 !       unsigned integer longword completion code
;  226     0357  1 !
;  227     0358  1 ! COMPLETION CODES:
;  228     0359  1 !
;  229     0360  1 !       STS$K_SEVERE (4) -      The specified command could not be executed
;  230     0361  1 !
;  231     0362  1 !       STS$K_SUCCESS (1) -     The specified command was executed
;  232     0363  1 !
;  233     0364  1 ! SIDE EFFECTS:
;  234     0365  1 !
;  235     0366  1 !       The semantic actions corresponding to the parsed DEBUG command are
;  236     0367  1 !       performed. Various states of the debugger and user program may be
;  237     0368  1 !       altered, and output may be displayed to the user and written to a log
;  238     0369  1 !       file.
;  239     0370  1 !
;  240     0371  1
;  241     0372  2     BEGIN
;  242     0373  2
;  243     0374  2     LOCAL
;  244     0375  2         VERB_NODE : REF DBG$VERB_NODE;          ! Command verb node
;  245     0376  2
;  246     0377  2
;  247     0378  2
;  248     0379  2     ! Check for a command to execute.
;  249     0380  2     !
;  250     0381  2     IF .VERB_NODE_PTR EQL 0 THEN RETURN STS$K_SUCCESS;
;  251     0382  2
;  252     0383  2
;  253     0384  2     ! Obtain the verb node and set the pointer to it to 0.
;  254     0385  2     !
;  255     0386  2     verb_node = ..verb_node_ptr;
```

```
;  256          0387  2      .verb_node_ptr = 0;
;  257          0388
;  258          0389
;  259          0390        ! Now transfer control to the appropriate subnetwork and return
;  260          0391        !
;  261          0392        RETURN
;  262          0393            ( CASE .VERB_NODE [DBG$B_VERB_LITERAL] FROM DBG$K_FIRST_VERB
;  263          0394                                                    TO DBG$K_LAST_VERB OF
;  264          0395                SET
;  265          0396
;  266          0397                [allocate_verb] :
;  267          0398                    dbg$nexecute_allocate (.verb_node, .message_vect);
;  268          0399
;  269          0400                [at_sign_verb] :
;  270          0401                    dbg$nexecute_at_sign (.verb_node, .message_vect);
;  271          0402
;  272          0403                [attach_verb] :
;  273          0404                    dbg$nexecute_attach (.verb_node, .message_vect);
;  274          0405
;  275          0406                [call_verb] :
;  276          0407                    dbg$nexecute_call (.verb_node, .message_vect);
;  277          0408
;  278          0409                [cancel_verb] :
;  279          0410                    dbg$nexecute_cancel (.verb_node, .message_vect);
;  280          0411
;  281          0412                [declare_verb] :
;  282          0413                    dbg$nexecute_declare (.verb_node, .message_vect);
;  283          0414
;  284          0415                [define_verb] :
;  285          0416                    dbg$nexecute_define (.verb_node, .message_vect);
;  286          0417
;  287          0418                [delete_verb] :
;  288          0419                    dbg$nexecute_delete (.verb_node, .message_vect);
;  289          0420
;  290          0421                [deposit_verb] :
;  291          0422                    (dbg$deposit(.verb_node);sts$k_success);
;  292          0423
;  293          0424                [DISPLAY_VERB]:
;  294          0425                    (DBG$SCR_EXECUTE_DISPLAY_CMD(.VERB_NODE, FALSE);
;  295          0426                     STS$K_SUCCESS);
;  296          0427
;  297          0428                [dump_verb] :
;  298          0429                    dbg$nexecute_dump(.verb_node, .message_vect);
;  299          0430
;  300          0431                [edit_verb] :
;  301          0432                    dbg$nexecute_edit(.verb_node, .message_vect);
;  302          0433
;  303          0434                [evaluate_verb] :
;  304          0435                    (dbg$evaluate(.verb_node);sts$k_success);
;  305          0436
;  306          0437                [examine_verb] :
;  307          0438                    (dbg$examine(.verb_node);sts$k_success);
;  308          0439
;  309          0440                [exit_verb] :
;  310          0441                    dbg$nexecute_exit (.verb_node, .message_vect);
;  311          0442
;  312          0443                [exitloop_verb] :
```

```
;   313          0444  !                dbg$nexecute_exitloop (.verb_node, .message_vect);
;   314          0445  !
;   315          0446  !            [for_verb] :
;   316          0447  !                dbg$nexecute_for (.verb_node, .message_vect);
;   317          0448  !
;   318          0449  !            [go_verb] :
;   319          0450  !                dbg$nexecute_go (.verb_node, .message_vect);
;   320          0451  !
;   321          0452  !            [help_verb] :
;   322          0453  !                dbg$nexecute_help (.verb_node, .message_vect);
;   323          0454  !
;   324          0455  !            [if_verb] :
;   325          0456  !                dbg$nexecute_if (.verb_node, .message_vect);
;   326          0457  !
;   327          0458  !            [repeat_verb] :
;   328          0459  !                dbg$nexecute_repeat (.verb_node, .message_vect);
;   329          0460  !
;   330          0461  !            [SAVE_VERB]:
;   331          0462  !                (DBG$SCR_EXECUTE_SAVE_CMD(.VERB_NODE);
;   332          0463  !                 STS$K_SUCCESS);
;   333          0464  !
;   334          0465  !            [SCROLL_VERB]:
;   335          0466  !                (DBG$SCR_EXECUTE_SCROLL_CMD(.VERB_NODE);
;   336          0467  !                 STS$K_SUCCESS);
;   337          0468  !
;   338          0469  !            [search_verb] :
;   339          0470  !                dbg$nexecute_search (.verb_node, .message_vect);
;   340          0471  !
;   341          0472  !            [SELECT_VERB]:
;   342          0473  !                (DBG$SCR_EXECUTE_SELECT_CMD(.VERB_NODE);
;   343          0474  !                 STS$K_SUCCESS);
;   344          0475  !
;   345          0476  !            [show_verb] :
;   346          0477  !                dbg$nexecute_show (.verb_node, .message_vect);
;   347          0478  !
;   348          0479  !            [set_verb] :
;   349          0480  !                dbg$nexecute_set (.verb_node, .message_vect);
;   350          0481  !
;   351          0482  !            [spawn_verb] :
;   352          0483  !                dbg$nexecute_spawn (.verb_node, .message_vect);
;   353          0484  !
;   354          0485  !            [step_verb] :
;   355          0486  !                dbg$nexecute_step (.verb_node, .message_vect);
;   356          0487  !
;   357          0488  !            [symbolize_verb] :
;   358          0489  !                dbg$nexecute_symbolize (.verb_node, .message_vect);
;   359          0490  !
;   360          0491  !            [type_verb] :
;   361          0492  !                dbg$nexecute_type (.verb_node, .message_vect);
;   362          0493  !
;   363          0494  !            [undefine_verb] :
;   364          0495  !                dbg$nexecute_undefine (.verb_node, .message_vect);
;   365          0496  !
;   366          0497  !            [while_verb] :
;   367          0498  !                dbg$nexecute_while (.verb_node, .message_vect);
;   368          0499  !
;   369          0500  !            [INRANGE, OUTRANGE] :
```

```
;  370          0501  4                    BEGIN
;  371          0502  4                    .message_vect = dbg$nmake_arg_vect (dbg$_notimplan, 1,
;  372          0503  4                                                      UPLIT BYTE (17, 'full verb support'));
;  373          0504  4                    sts$k_severe
;  374          0505  3                    END;
;  375          0506  3
;  376          0507  2                TES );
;  377          0508  1    END;


                                                        .TITLE  DBGNEXCTE
                                                        .IDENT  \V04-000\

                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                                    11  00000 P.AAA:  .BYTE   17
6F  70  70  75  73  20  62  72  65  76  20  6C  6C  75  66  00001      .ASCII  \full verb support\                    ;
                                                    74  72  00010                                                  ;

                                                        .EXTRN  DBG$DEF_PR_EXIT
                                                        .EXTRN  DBG$DEF_SYM_ADD
                                                        .EXTRN  DBG$DEF_SYM_FIND
                                                        .EXTRN  DBG$DEPOSIT, DBG$EVALUATE
                                                        .EXTRN  DBG$EXAMINE, DBG$GET_MEMORY
                                                        .EXTRN  DBG$GET_TEMPMEM
                                                        .EXTRN  DBG$MAKE_VMS_DESC
                                                        .EXTRN  DBG$NCOPY_DESC, DBG$NEXECUTE_ALLOCATE
                                                        .EXTRN  DBG$NEXECUTE_AT_SIGN
                                                        .EXTRN  DBG$NEXECUTE_ATTACH
                                                        .EXTRN  DBG$NEXECUTE_CALL
                                                        .EXTRN  DBG$NEXECUTE_CANCEL
                                                        .EXTRN  DBG$NEXECUTE_DECLARE
                                                        .EXTRN  DBG$NEXECUTE_DEFINE
                                                        .EXTRN  DBG$NEXECUTE_DELETE
                                                        .EXTRN  DBG$NEXECUTE_DUMP
                                                        .EXTRN  DBG$NEXECUTE_EDIT
                                                        .EXTRN  DBG$NEXECUTE_EXIT
                                                        .EXTRN  DBG$NEXECUTE_EXITLOOP
                                                        .EXTRN  DBG$NEXECUTE_FOR
                                                        .EXTRN  DBG$NEXECUTE_GO
                                                        .EXTRN  DBG$NEXECUTE_HELP
                                                        .EXTRN  DBG$NEXECUTE_IF
                                                        .EXTRN  DBG$NEXECUTE_REPEAT
                                                        .EXTRN  DBG$NEXECUTE_SEARCH
                                                        .EXTRN  DBG$NEXECUTE_SET
                                                        .EXTRN  DBG$NEXECUTE_SHOW
                                                        .EXTRN  DBG$NEXECUTE_SPAWN
                                                        .EXTRN  DBG$NEXECUTE_STEP
                                                        .EXTRN  DBG$NEXECUTE_SYMBOLIZE
                                                        .EXTRN  DBG$NEXECUTE_TYPE
                                                        .EXTRN  DBG$NEXECUTE_UNDEFINE
                                                        .EXTRN  DBG$NEXECUTE_WHILE
                                                        .EXTRN  DBG$NFREE_DESC, DBG$NGET_LVAL
                                                        .EXTRN  DBG$NGET_SYMID, DBG$NGET_TYPE
                                                        .EXTRN  DBG$NMAKE_ARG_VECT
                                                        .EXTRN  DBG$NOUT_INFO, DBG$REL_MEMORY
                                                        .EXTRN  DBG$SCR_EXECUTE_DISPLAY_CMD
```

```
                                                        .EXTRN  DBG$SCR_EXECUTE_SAVE_CMD
                                                        .EXTRN  DBG$SCR_EXECUTE_SCROLL_CMD
                                                        .EXTRN  DBG$SCR_EXECUTE_SELECT_CMD
                                                        .EXTRN  DBG$STA_LOCK_SYMID
                                                        .EXTRN  DBG$GL_CISHEAD, DBG$GL_CIS_LEVELS
                                                        .EXTRN  DBG$GB_DEF_OUT, DBG$GL_SCREEN_ERROR
                                                        .EXTRN  DBG$GL_SCREEN_NOGO
                                                        .EXTRN  DBG$GL_SCREEN_OUTPUT
                                                        .EXTRN  DBG$GL_SCREEN_SOURCE

                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                        0004 00000      .ENTRY  DBG$NEXECUTE_CMD, Save R2                               ; 0330
                              50    04  AC  D0 00002    MOVL    VERB_NODE_PTR, R0                                       ; 0381
                                        03  12 00006    BNEQ    1$
                                    019F  31 00008    BRW     30$
                              52        60  D0 0000B 1$:  MOVL    (R0), VERB_NODE                                 ; 0386
                                        60  D4 0000E    CLRL    (R0)                                                    ; 0387
                           20           62  8F 00010    CASEB   (VERB_NODE), #1, #32                                    ; 0393
                  00AD     0093     0086     006C  00014 2$:  .WORD   4$-2$,-
                  010F     0104     00F9     00C7  0001C              6$-2$,-
                  01C1     019A     01A7     0136  00024              7$-2$,-
                  0150     0180     01DB     0143  0002C              9$-2$,-
                  00A0     011C     015D     01F5  00034              11$-2$,-
                  01E8     01CE     005F     0184  0003C              15$-2$,-
                  00D2     00DF     0079     0129  00044              16$-2$,-
                  00BA     016A     018D     0175  0004C              18$-2$,-
                                             00EC  00054              21$-2$,-
                                                                      32$-2$,-
                                                                      31$-2$,-
                                                                      34$-2$,-
                                                                      22$-2$,-
                                                                      36$-2$,-
                                                                      28$-2$,-
                                                                      23$-2$,-
                                                                      38$-2$,-
                                                                      24$-2$,-
                                                                      19$-2$,-
                                                                      8$-2$,-
                                                                      33$-2$,-
                                                                      3$-2$,-
                                                                      35$-2$,-
                                                                      37$-2$,-
                                                                      20$-2$,-
                                                                      5$-2$,-
                                                                      13$-2$,-
                                                                      12$-2$,-
                                                                      26$-2$,-
                                                                      29$-2$,-
                                                                      25$-2$,-
                                                                      10$-2$,-
                                                                      14$-2$
                        00000000'   EF  9F 00056        PUSHAB  P.AAA                                                   ; 0503
                                    01  DD 0005C        PUSHL   #1                                                      ; 0502
                        00028250    8F  DD 0005E        PUSHL   #164432
                  00000000G   00    03  FB 00064        CALLS   #3, DBG$NMAKE_ARG_VECT
                        08    BC    50  D0 0006B        MOVL    R0, @MESSAGE_VECT
```

```
                     50              04 DD 0006F          MOVL    #4, R0                                  ; 0501
                                     04 00072             RET
                     08 AC           DD 00073 3$:         PUSHL   MESSAGE_VECT                            ; 0398
                        52           DD 00076             PUSHL   VERB_NODE
          00000000G  00              02 FB 00078          CALLS   #2, DBG$NEXECUTE_ALLOCATE
                                     04 0007F             RET
                     08 AC           DD 00080 4$:         PUSHL   MESSAGE_VECT                            ; 0401
                        52           DD 00083             PUSHL   VERB_NODE
          00000000G  00              02 FB 00085          CALLS   #2, DBG$NEXECUTE_AT_SIGN
                                     04 0008C             RET
                     08 AC           DD 0008D 5$:         PUSHL   MESSAGE_VECT                            ; 0404
                        52           DD 00090             PUSHL   VERB_NODE
          00000000G  00              02 FB 00092          CALLS   #2, DBG$NEXECUTE_ATTACH
                                     04 00099             RET
                     08 AC           DD 0009A 6$:         PUSHL   MESSAGE_VECT                            ; 0407
                        52           DD 0009D             PUSHL   VERB_NODE
          00000000G  00              02 FB 0009F          CALLS   #2, DBG$NEXECUTE_CALL
                                     04 000A6             RET
                     08 AC           DD 000A7 7$:         PUSHL   MESSAGE_VECT                            ; 0410
                        52           DD 000AA             PUSHL   VERB_NODE
          00000000G  00              02 FB 000AC          CALLS   #2, DBG$NEXECUTE_CANCEL
                                     04 000B3             RET
                     08 AC           DD 000B4 8$:         PUSHL   MESSAGE_VECT                            ; 0413
                        52           DD 000B7             PUSHL   VERB_NODE
          00000000G  00              02 FB 000B9          CALLS   #2, DBG$NEXECUTE_DECLARE
                                     04 000C0             RET
                     08 AC           DD 000C1 9$:         PUSHL   MESSAGE_VECT                            ; 0416
                        52           DD 000C4             PUSHL   VERB_NODE
          00000000G  00              02 FB 000C6          CALLS   #2, DBG$NEXECUTE_DEFINE
                                     04 000CD             RET
                     08 AC           DD 000CE 10$:        PUSHL   MESSAGE_VECT                            ; 0419
                        52           DD 000D1             PUSHL   VERB_NODE
          00000000G  00              02 FB 000D3          CALLS   #2, DBG$NEXECUTE_DELETE
                                     04 000DA             RET
                        52           DD 000DB 11$:        PUSHL   VERB_NODE                               ; 0422
          00000000G  00              01 FB 000DD          CALLS   #1, DBG$DEPOSIT
                                     3B 11 000E4          BRB     17$
                                     7E D4 000E6 12$:     CLRL    -(SP)                                   ; 0425
                        52           DD 000E8             PUSHL   VERB_NODE
          00000000G  00              02 FB 000EA          CALLS   #2, DBG$SCR_EXECUTE_DISPLAY_CMD
                                     2E 11 000F1          BRB     17$
                     08 AC           DD 000F3 13$:        PUSHL   MESSAGE_VECT                            ; 0429
                        52           DD 000F6             PUSHL   VERB_NODE
          00000000G  00              02 FB 000F8          CALLS   #2, DBG$NEXECUTE_DUMP
                                     04 000FF             RET
                     08 AC           DD 00100 14$:        PUSHL   MESSAGE_VECT                            ; 0432
                        52           DD 00103             PUSHL   VERB_NODE
          00000000G  00              02 FB 00105          CALLS   #2, DBG$NEXECUTE_EDIT
                                     04 0010C             RET
                        52           DD 0010D 15$:        PUSHL   VERB_NODE                               ; 0435
          00000000G  00              01 FB 0010F          CALLS   #1, DBG$EVALUATE
                                     7A 11 00116          BRB     27$
                        52           DD 00118 16$:        PUSHL   VERB_NODE                               ; 0438
          00000000G  00              01 FB 0011A          CALLS   #1, DBG$EXAMINE
                                     6F 11 00121 17$:     BRB     27$
                     08 AC           DD 00123 18$:        PUSHL   MESSAGE_VECT                            ; 0441
                        52           DD 00126             PUSHL   VERB_NODE
```

```
                00000000G  00        02  FB 00128          CALLS   #2, DBG$NEXECUTE_EXIT                          ;
                                      04 0012F           RET                                                    ;
                           08     AC  DD 00130 19$:      PUSHL   MESSAGE_VECT                                   ; 0444
                                  52  DD 00133           PUSHL   VERB_NODE                                      ;
                00000000G  00        02  FB 00135          CALLS   #2, DBG$NEXECUTE_EXITLOOP                      ;
                                      04 0013C           RET                                                    ;
                           08     AC  DD 0013D 20$:      PUSHL   MESSAGE_VECT                                   ; 0447
                                  52  DD 00140           PUSHL   VERB_NODE                                      ;
                00000000G  00        02  FB 00142          CALLS   #2, DBG$NEXECUTE_FOR                           ;
                                      04 00149           RET                                                    ;
                           08     AC  DD 0014A 21$:      PUSHL   MESSAGE_VECT                                   ; 0450
                                  52  DD 0014D           PUSHL   VERB_NODE                                      ;
                00000000G  00        02  FB 0014F          CALLS   #2, DBG$NEXECUTE_GO                            ;
                                      04 00156           RET                                                    ;
                           08     AC  DD 00157 22$:      PUSHL   MESSAGE_VECT                                   ; 0453
                                  52  DD 0015A           PUSHL   VERB_NODE                                      ;
                00000000G  00        02  FB 0015C          CALLS   #2, DBG$NEXECUTE_HELP                          ;
                                      04 00163           RET                                                    ;
                           08     AC  DD 00164 23$:      PUSHL   MESSAGE_VECT                                   ; 0456
                                  52  DD 00167           PUSHL   VERB_NODE                                      ;
                00000000G  00        02  FB 00169          CALLS   #2, DBG$NEXECUTE_IF                            ;
                                      04 00170           RET                                                    ;
                           08     AC  DD 00171 24$:      PUSHL   MESSAGE_VECT                                   ; 0459
                                  52  DD 00174           PUSHL   VERB_NODE                                      ;
                00000000G  00        02  FB 00176          CALLS   #2, DBG$NEXECUTE_REPEAT                        ;
                                      04 0017D           RET                                                    ;
                                  52  DD 0017E 25$:      PUSHL   VERB_NODE                                      ; 0462
                00000000G  00        01  FB 00180          CALLS   #1, DBG$SCR_EXECUTE_SAVE_CMD                   ;
                                      21  11 00187          BRB     30$                                            ;
                                  52  DD 00189 26$:      PUSHL   VERB_NODE                                      ; 0466
                00000000G  00        01  FB 00188          CALLS   #1, DBG$SCR_EXECUTE_SCROLL_CMD                 ;
                                      16  11 00192 27$:      BRB     30$                                            ;
                           08     AC  DD 00194 28$:      PUSHL   MESSAGE_VECT                                   ; 0470
                                  52  DD 00197           PUSHL   VERB_NODE                                      ;
                00000000G  00        02  FB 00199          CALLS   #2, DBG$NEXECUTE_SEARCH                        ;
                                      04 001A0           RET                                                    ;
                                  52  DD 001A1 29$:      PUSHL   VERB_NODE                                      ; 0473
                00000000G  00        01  FB 001A3          CALLS   #1, DBG$SCR_EXECUTE_SELECT_CMD                 ;
                           50         01  D0 001AA 30$:      MOVL    #1, R0                                         ;
                                      04 001AD           RET                                                    ;
                           08     AC  DD 001AE 31$:      PUSHL   MESSAGE_VECT                                   ; 0477
                                  52  DD 001B1           PUSHL   VERB_NODE                                      ;
                00000000G  00        02  FB 001B3          CALLS   #2, DBG$NEXECUTE_SHOW                          ;
                                      04 001BA           RET                                                    ;
                           08     AC  DD 001BB 32$:      PUSHL   MESSAGE_VECT                                   ; 0480
                                  52  DD 001BE           PUSHL   VERB_NODE                                      ;
                00000000G  00        02  FB 001C0          CALLS   #2, DBG$NEXECUTE_SET                           ;
                                      04 001C7           RET                                                    ;
                           08     AC  DD 001C8 33$:      PUSHL   MESSAGE_VECT                                   ; 0483
                                  52  DD 001CB           PUSHL   VERB_NODE                                      ;
                00000000G  00        02  FB 001CD          CALLS   #2, DBG$NEXECUTE_SPAWN                         ;
                                      04 001D4           RET                                                    ;
                           08     AC  DD 001D5 34$:      PUSHL   MESSAGE_VECT                                   ; 0486
                                  52  DD 001D8           PUSHL   VERB_NODE                                      ;
                00000000G  00        02  FB 001DA          CALLS   #2, DBG$NEXECUTE_STEP                          ;
                                      04 001E1           RET                                                    ;
                           08     AC  DD 001E2 35$:      PUSHL   MESSAGE_VECT                                   ; 0489
```

```
                                        52  DD 001E5           PUSHL   VERB_NODE                                          ;
                00000000G   00          02  FB 001E7           CALLS   #2, DBG$NEXECUTE_SYMBOLIZE                       ;
                                        04  001EE             RET                                                      ;
                                    08  AC  DD 001EF 36$:      PUSHL   MESSAGE_VECT                                     ; 0492
                                        52  DD 001F2           PUSHL   VERB_NODE                                        ;
                00000000G   00          02  FB 001F4           CALLS   #2, DBG$NEXECUTE_TYPE                            ;
                                        04  001FB             RET                                                      ;
                                    08  AC  DD 001FC 37$:      PUSHL   MESSAGE_VECT                                     ; 0495
                                        52  DD 001FF           PUSHL   VERB_NODE                                        ;
                00000000G   00          02  FB 00201           CALLS   #2, DBG$NEXECUTE_UNDEFINE                        ;
                                        04  00208             RET                                                      ;
                                    08  AC  DD 00209 38$:      PUSHL   MESSAGE_VECT                                     ; 0498
                                        52  DD 0020C           PUSHL   VERB_NODE                                        ;
                00000000G   00          02  FB 0020E           CALLS   #2, DBG$NEXECUTE_WHILE                           ;
                                        04  00215             RET                                                      ; 0508

; Routine Size:  534 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  379          0509  1 GLOBAL ROUTINE DBG$NCIS_ADD (POINTER, LENGTH, TYPE,
;  380          0510  1                                REPEAT_COUNT, WHILE_CLAUSE, LOOP_INCR) =
;  381          0511  1 !
;  382          0512  1 ! FUNCTION
;  383          0513  1 !       This routine creates and adds a new Command Input Stream (CIS) Entry
;  384          0514  1 !       to the Command Input Stream Stack.  The global variable DBG$GL_CISHEAD
;  385          0515  1 !       is set to point to the new CIS Entry so that DEBUG commands are gotten
;  386          0516  1 !       from this new CIS Entry first.  The forward link in the new entry is
;  387          0517  1 !       set to contain the old value of DBG$GL_CISHEAD so that the previous
;  388          0518  1 !       CIS entry is restored once the new CIS entry is emptied of commands.
;  389          0519  1 !
;  390          0520  1 ! INPUTS
;  391          0521  1 !       POINTER - The address of either a buffer or a RAB to be placed
;  392          0522  1 !                 in the DSC$A_POINTER field of the new link.
;  393          0523  1 !
;  394          0524  1 !       LENGTH  - The length of the above buffer (0 for RAB).
;  395          0525  1 !
;  396          0526  1 !       TYPE    - The type of the link to be added.
;  397          0527  1 !
;  398          0528  1 !       REPEAT_COUNT - The count for a CIS of type CIS_REPEAT.  For a CIS of
;  399          0529  1 !                 type FOR, this contains the upper bound.
;  400          0530  1 !
;  401          0531  1 !       WHILE_CLAUSE - A counted string with the action clause for a CIS of
;  402          0532  1 !                 type CIS_WHILE.  For a CIS of type FOR, this contains the
;  403          0533  1 !                 name of the loop variable.
;  404          0534  1 !
;  405          0535  1 !       LOOP_INCR - The loop increment in FOR loops.
;  406          0536  1 !
;  407          0537  1 !
;  408          0538  1 ! OUTPUTS
;  409          0539  1 !       This routine returns STS$K_SUCCESS as its value.
;  410          0540  1 !
;  411          0541  1
;  412          0542  2     BEGIN
;  413          0543  2
;  414          0544  2     MAP
;  415          0545  2         WHILE_CLAUSE: REF VECTOR [,BYTE];
;  416          0546  2
;  417          0547  2     LOCAL
;  418          0548  2         FOR_LOOP_VAR,                   ! Points to counted string with FOR
;  419          0549  2                                         !      loop variable
;  420          0550  2         FOR_UPPER_BOUND,                ! Integer with upper bound for FOR loops
;  421          0551  2         TEMP;                           ! Temporary pointer to head CIS node
;  422          0552  2
;  423          0553  2
;  424          0554  2
;  425          0555  2     ! Increment the count of the number of levels of CIS we have.
;  426          0556  2     !
;  427          0557  2     DBG$GL_CIS_LEVELS = .DBG$GL_CIS_LEVELS + 1;
;  428          0558  2
;  429          0559  2
;  430          0560  2     ! Pick up the FOR-loop bounds if this is a FOR-loop CIS.
;  431          0561  2     !
;  432          0562  2     FOR_LOOP_VAR = .WHILE_CLAUSE;
;  433          0563  2     FOR_UPPER_BOUND = .REPEAT_COUNT;
;  434          0564  2
;  435          0565  2
```

```
;  436          0566  2      ! Save current list head and allocate a new one
;  437          0567  2      !
;  438          0568  2      TEMP = .DBG$GL_CISHEAD ;
;  439          0569  2      DBG$GL_CISHEAD = DBG$GET_MEMORY ((CIS_ELEMENTS+3)/%UPVAL);
;  440          0570  2      DBG$GL_CISHEAD [CIS$A_NEXT_LINK] = .TEMP;
;  441          0571  2      DBG$GL_CISHEAD [CIS$A_INPUT_PTR] = .POINTER;
;  442          0572  2      DBG$GL_CISHEAD [CIS$B_INPUT_TYPE] = .TYPE;
;  443          0573  2      DBG$GL_CISHEAD [CIS$W_LENGTH]     = .LENGTH;
;  444          0574  2
;  445          0575  2      IF .TYPE EQL CIS_REPEAT
;  446          0576  2      THEN
;  447          0577  2          DBG$GL_CISHEAD [CIS$L_REPEAT_COUNT] = .REPEAT_COUNT;
;  448          0578  2
;  449          0579  2      IF .TYPE EQL CIS_WHILE
;  450          0580  2      THEN
;  451          0581  2          DBG$GL_CISHEAD [CIS$V_WHILE_FLAG] = .WHILE_CLAUSE;
;  452          0582  2
;  453          0583  2      IF .TYPE EQL CIS_FOR
;  454          0584  2      THEN
;  455          0585  3          BEGIN
;  456          0586  3          DBG$GL_CISHEAD [CIS$L_FOR_UPPER_BOUND] = .FOR_UPPER_BOUND;
;  457          0587  3          DBG$GL_CISHEAD [CIS$A_FOR_LOOP_VAR] = .FOR_LOOP_VAR;
;  458          0588  3          DBG$GL_CISHEAD [CIS$L_FOR_LOOP_INCR] = .LOOP_INCR;
;  459          0589  3          END;
;  460          0590  2
;  461          0591  2
;  462          0592  2      ! The fields INIT_ADDR and INIT_LENGTH are used to determine
;  463          0593  2      ! how much storage to release for this buffer, since the pointer
;  464          0594  2      ! field is modified by the parser among others.
;  465          0595  2      !
;  466          0596  2      DBG$GL_CISHEAD [CIS$A_INIT_ADDR]    = .POINTER;
;  467          0597  2
;  468          0598  2
;  469          0599  2      ! If we are adding an input buffer add 1 byte to the length
;  470          0600  2      ! to be released because we allocated an extra one so we could
;  471          0601  2      ! guarantee a zero byte at the end of the string.
;  472          0602  2      !
;  473          0603  2      IF .TYPE EQL CIS_INPBUF
;  474          0604  2      THEN
;  475          0605  2          DBG$GL_CISHEAD [CIS$W_INIT_LENGTH]      = .LENGTH + 1
;  476          0606  2
;  477          0607  2      ELSE
;  478          0608  2          DBG$GL_CISHEAD [CIS$W_INIT_LENGTH]      = .LENGTH;
;  479          0609  2
;  480          0610  2      RETURN STS$K_SUCCESS;
;  481          0611  2
;  482          0612  1      END;


                                 003C 00000          .ENTRY  DBG$NCIS_ADD, Save R2,R3,R4,R5                  ; 0509
                55 00000000G   00  9E 00002          MOVAB   DBG$GL_CISHEAD, R5                              ;
                   00000000G   00  D6 00009          INCL    DBG$GL_CIS_LEVELS                               ; 0557
                53         10   AC  7D 0000F          MOVQ    REPEAT_COUNT, FOR_UPPER_BOUND                   ; 0563
                52              65  D0 00013          MOVL    DBG$GL_CISHEAD, TEMP                            ; 0568
```

```
                                                    0E  DD 00016             PUSHL   #14                                             ; 0569
                                 00000000G      00  01  FB 00018             CALLS   #1, DBG$GET_MEMORY                              ;
                                                65  50  D0 0001F             MOVL    R0, DBG$GL_CISHEAD                              ;
                                          08    A0  52  D0 00022             MOVL    TEMP, 8(R0)                                     ; 0570
                                          04    A0  AC  D0 00026             MOVL    POINTER, 4(R0)                                  ; 0571
                                                51  0C  AC  D0 0002B         MOVL    TYPE, R1                                        ; 0572
                                          02    A0  51  90 0002F             MOVB    R1, 2(R0)                                       ;
                                                60  08  AC  B0 00033         MOVW    LENGTH, (R0)                                    ; 0573
                                                04  51  D1 00037             CMPL    R1, #4                                          ; 0575
                                                    05  12 0003A             BNEQ    1$                                              ;
                                          18    A0  10  AC  D0 0003C         MOVL    REPEAT_COUNT, 24(R0)                            ; 0577
                                                05  51  D1 00041 1$:         CMPL    R1, #5                                          ; 0579
                                                    07  12 00044             BNEQ    2$                                              ;
                 12   A0                  01            01  14  AC  F0 00046 INSV    WHILE_CLAUSE, #1, #1, 18(R0)                    ; 0581
                                                07  51  D1 0004D 2$:         CMPL    R1, #7                                          ; 0583
                                                    09  12 00050             BNEQ    3$                                              ;
                                          18    A0  53  7D 00052             MOVQ    FOR_UPPER_BOUND, 24(R0)                         ; 0586
                                          20    A0  18  AC  D0 00056         MOVL    LOOP_INCR, 32(R0)                               ; 0588
                                          0C    A0  04  AC  D0 0005B 3$:     MOVL    POINTER, 12(R0)                                 ; 0596
                                                02  51  D1 00060             CMPL    R1, #2                                          ; 0603
                                                    08  12 00063             BNEQ    4$                                              ;
                             10   A0          08    AC  01  A1 00065         ADDW3   #1, LENGTH, 16(R0)                              ; 0605
                                                    05  11 0006B             BRB     5$                                              ;
                                          10    A0  08  AC  B0 0006D 4$:     MOVW    LENGTH, 16(R0)                                  ; 0608
                                                50  01  D0 00072 5$:         MOVL    #1, R0                                          ; 0610
                                                    04 00075                 RET                                                     ; 0612

; Routine Size:  118 bytes,    Routine Base:  DBG$CODE + 0216
```

```
;  484     0613  1 GLOBAL ROUTINE DBG$NCIS_OPENICF (MESSAGE_VECT) =
;  485     0614  1 !++
;  486     0615  1 ! FUNCTIONAL DESCRIPTION:
;  487     0616  1 !       Routine is called when there is a RAB at the top of the command
;  488     0617  1 !       input stream. It opens the related FAB and connects the RAB to it
;  489     0618  1 !
;  490     0619  1 ! FORMAL PARAMETERS:
;  491     0620  1 !
;  492     0621  1 !       message_vect    - address of a longword to contain address of message vector
;  493     0622  1 !
;  494     0623  1 ! IMPLICIT INPUTS:
;  495     0624  1 !       The head of the command input stream
;  496     0625  1 !
;  497     0626  1 ! IMPLICIT OUTPUTS:
;  498     0627  1 !
;  499     0628  1 !       on failure, a message argument vector
;  500     0629  1 !
;  501     0630  1 ! ROUTINE VALUE:
;  502     0631  1 !
;  503     0632  1 !       sts$k_success (1) - action performed
;  504     0633  1 !
;  505     0634  1 !       sts$k_severe (4) - failure
;  506     0635  1 !
;  507     0636  1 ! SIDE EFFECTS:
;  508     0637  1 !       A FAB is opened and a RAB connected to it. If SET OUTPUT VERIFY, then
;  509     0638  1 !       a message is generated indicating we are entering an indirect command file
;  510     0639  1 !--
;  511     0640  2     BEGIN
;  512     0641  2
;  513     0642  2     LOCAL
;  514     0643  2         STATUS,                         ! Holds RMS status code
;  515     0644  2         FAB_PTR : REF $FAB_DECL,        ! File access block pointer
;  516     0645  2         RAB_PTR : REF $RAB_DECL;        ! Record access block pointer
;  517     0646  2
;  518     0647  2     ! Extract the related FAB from the RAB at the top of the cis
;  519     0648  2     !
;  520     0649  2     rab_ptr = .dbg$gl_cishead [cis$a_input_ptr];
;  521     0650  2     fab_ptr = .rab_ptr [rab$l_fab];
;  522     0651  2
;  523     0652  2     status = $OPEN (FAB=.fab_ptr);
;  524     0653  2     IF NOT .status
;  525     0654  2     THEN
;  526     0655  2         BEGIN
;  527     0656  2
;  528     0657  2         LOCAL
;  529     0658  2                 MSG_DESC : REF dbg$stg_desc;    ! String descriptor for message
;  530     0659  2
;  531     0660  2         msg_desc = dbg$get_tempmem (2);
;  532     0661  2
;  533     0662  2         msg_desc[dsc$w_length]  = .fab_ptr[fab$b_fns];
;  534     0663  2         msg_desc[dsc$a_pointer] = .fab_ptr[fab$l_fna];
;  535     0664  2
;  536     0665  2
;  537     0666  2         ! Flag link for removal so we won't try to read from it again
;  538     0667  2         !
;  539     0668  2         dbg$gl_cishead[cis$v_rem_flag] = 1;
;  540     0669  2
```

```
;  541          0670  3          .message_vect = dbg$nmake_arg_vect (shr$_openin + dbg_fac_code,
;  542          0671  3                                              1,
;  543          0672  3                                             .msg_desc, .fab_ptr[fab$l_sts], .fab_ptr[fab$l_stv]);
;  544          0673  3
;  545          0674  3          RETURN sts$k_severe;
;  546          0675  3
;  547          0676  3          END;
;  548          0677  3
;  549          0678  3
;  550          0679  3      ! Connect the RAB to the just opened FAB
;  551          0680  3      !
;  552          0681  3      status = $CONNECT (RAB=.rab_ptr);
;  553          0682  3      IF NOT .status
;  554          0683  3      THEN
;  555          0684  3          BEGIN
;  556          0685  3          LOCAL
;  557          0686  3                  MSG_DESC : REF dbg$stg_desc; ! string descriptor for message
;  558          0687  3
;  559          0688  3          msg_desc = dbg$get_tempmem (2);
;  560          0689  3
;  561          0690  3          msg_desc[dsc$w_length]  = .fab_ptr[fab$b_fns];
;  562          0691  3          msg_desc[dsc$a_pointer] = .fab_ptr[fab$l_fna];
;  563          0692  3
;  564          0693  3
;  565          0694  3          ! Flag link for removal so we won't try to read from it again
;  566          0695  3          !
;  567          0696  3          dbg$gl_cishead[cis$v_rem_flag] = 1;
;  568          0697  3
;  569          0698  3          .message_vect = dbg$nmake_arg_vect (shr$_openin + dbg_fac_code,
;  570          0699  3                                              1, .msg_desc,
;  571          0700  3                                              .fab_ptr[fab$l_sts],
;  572          0701  3                                              .fab_ptr[fab$l_stv]);
;  573          0702  3
;  574          0703  3          RETURN sts$k_severe;
;  575          0704  3
;  576          0705  3          END;
;  577          0706  3
;  578          0707  3
;  579          0708  3      ! Check for verification message.
;  580          0709  3      !
;  581          0710  3      IF .dbg$gb_def_out [out_verify]
;  582          0711  3      THEN
;  583          0712  3          icf_message(1);
;  584          0713  3
;  585          0714  3      RETURN sts$k_success;
;  586          0715  3
;  587          0716  1      END;


                                                    .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                          08  00012 P.AAB:  .BYTE   8
;                 67  6E  69  72  65  74  6E  65  00013          .ASCII  \entering\
;                                          07  0001B P.AAC:  .BYTE   7
;                     67  6E  69  74  69  78  65  0001C          .ASCII  \exiting\
```

```
                                                     ENTER_PHRASE=       P.AAB
                                                     EXIT_PHRASE=        P.AAC
                                                             .EXTRN  SYS$OPEN, SYS$CONNECT

                                                             .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                 003C 00000          .ENTRY  DBG$NCIS_OPENICF, Save R2,R3,R4,R5      ; 0613
                                      55 00000000G   00   9E 00002          MOVAB   DBG$GL_CISHEAD, R5              ;
                                      50             65   D0 00009          MOVL    DBG$GL_CISHEAD, R0              ; 0649
                                      53         04  A0   D0 0000C          MOVL    4(R0), RAB_PTR                  ;
                                      52         3C  A3   D0 00010          MOVL    60(RAB_PTR), FAB_PTR            ; 0650
                                                 52   DD 00014          PUSHL   FAB_PTR                         ; 0652
                                00000000G        00   FB 00016          CALLS   #1, SYS$OPEN                    ;
                                      54             50   D0 0001D          MOVL    R0, STATUS                      ;
                                      0F             54   E9 00020          BLBC    STATUS, 1$                      ; 0653
                                                     53   DD 00023          PUSHL   RAB_PTR                         ; 0681
                                00000000G        00   FB 00025          CALLS   #1, SYS$CONNECT                 ;
                                      54             50   D0 0002C          MOVL    R0, STATUS                      ;
                                      36             54   E8 0002F          BLBS    STATUS, 2$                      ; 0682
                                                     02   DD 00032  1$:      PUSHL   #2                              ; 0688
                                00000000G        00   FB 00034          CALLS   #1, DBG$GET_TEMPMEM             ;
                                      60         34  A2   9B 0003B          MOVZBW  52(FAB_PTR), (MSG_DESC)         ; 0690
                                04    A0         2C  A2   D0 0003F          MOVL    44(FAB_PTR), 4(MSG_DESC)        ; 0691
                                      51             65   D0 00044          MOVL    DBG$GL_CISHEAD, R1              ; 0696
                                12    A1             01   88 00047          BISB2   #1, 18(R1)                      ;
                                      7E         08  A2   7D 0004B          MOVQ    8(FAB_PTR), -(SP)               ; 0700
                                                     50   DD 0004F          PUSHL   MSG_DESC                        ; 0699
                                                     01   DD 00051          PUSHL   #1                              ; 0698
                                          00021098   8F   DD 00053          PUSHL   #135320                         ;
                                00000000G        00   05   FB 00059          CALLS   #5, DBG$NMAKE_ARG_VECT          ;
                                04    BC             50   D0 00060          MOVL    R0, @MESSAGE_VECT               ;
                                      50             04   D0 00064          MOVL    #4, R0                          ; 0703
                                                     04   00067          RET                                     ;
                                      1F 00000000G   00   E9 00068  2$:      BLBC    DBG$GB_DEF_OUT+2, 3$            ; 0710
                                      50 00000000'   EF   9E 0006F          MOVAB   ENTER_PHRASE, PHRASE            ; 0712
                                             2C      A2   DD 00076          PUSHL   44(FAB_PTR)                     ;
                                      7E         34  A2   9A 00079          MOVZBL  52(FAB_PTR), -(SP)              ;
                                                     50   DD 0007D          PUSHL   PHRASE                          ;
                                                     03   DD 0007F          PUSHL   #3                              ;
                                          0002808B   8F   DD 00081          PUSHL   #163979                         ;
                                00000000G        00   05   FB 00087          CALLS   #5, DBG$NOUT_INFO               ;
                                      50             01   D0 0008E  3$:      MOVL    #1, R0                          ; 0714
                                                     04   00091          RET                                     ; 0716

; Routine Size:  146 bytes,    Routine Base:  DBG$CODE + 028C

;  588          0717  1
```

```
;  590     0718  1 GLOBAL ROUTINE DBG$NCIS_REMOVE(EXIT_FLAG, MESSAGE_VECT) =
;  591     0719  1 !
;  592     0720  1 ! FUNCTIONAL DESCRIPTION:
;  593     0721  1 !       Removes the top link from the command input stream and delete the
;  594     0722  1 !       storage for it. If the link has additional dynamic storage related to
;  595     0723  1 !       it, such as a FAB,RAB, input buffer etc., that storage is freed also.
;  596     0724  1 !
;  597     0725  1 ! FORMAL PARAMETERS:
;  598     0726  1 !
;  599     0727  1 !       EXIT_FLAG       - TRUE if this routine is called from EXIT or EXITLOOP.
;  600     0728  1 !
;  601     0729  1 !       MESSAGE_VECT    - The address of a longword to contain the address of
;  602     0730  1 !                         a message argument vector.
;  603     0731  1 !
;  604     0732  1 ! IMPLICIT INPUTS:
;  605     0733  1 !
;  606     0734  1 !       The head of the command input stream.
;  607     0735  1 !
;  608     0736  1 ! IMPLICIT OUTPUTS:
;  609     0737  1 !
;  610     0738  1 !       On error, a message argument vector is constructed and returned.
;  611     0739  1 !
;  612     0740  1 ! ROUTINE VALUE:
;  613     0741  1 !
;  614     0742  1 !       STS$K_SUCCESS (1) - Success. Actions performed.
;  615     0743  1 !
;  616     0744  1 !       STS$K_SEVERE (4) - Failure. Error message argument vector constructed.
;  617     0745  1 !
;  618     0746  1 ! SIDE EFFECTS:
;  619     0747  1 !       The head of the command input stream is reset to what was the
;  620     0748  1 !       "next" link before this routine was called. If SET OUTPUT VERIFY,
;  621     0749  1 !       then a message is generated saying we are exiting the indirect
;  622     0750  1 !       command file.
;  623     0751  1 !
;  624     0752  1
;  625     0753  2     BEGIN
;  626     0754  2
;  627     0755  2     LOCAL
;  628     0756  2         BOUNDS_MATCH,               ! TRUE when FOR loop lower bound matches upper bound
;  629     0757  2         BUFLIST: REF VECTOR[],      !
;  630     0758  2         COND,                       ! TRUE or FALSE: condition value in WHILE cis
;  631     0759  2         DUMMY,                      ! dummy output parameter
;  632     0760  2         GLOBAL_FLAG,                ! output param for DEF_SYM_FIND
;  633     0761  2         KIND,                       ! kind of define symbol
;  634     0762  2         LOOP_INCR,                  ! the loop increment
;  635     0763  2         NEW_NAME,                   ! Pointer to the loop variable name
;  636     0764  2         NEW_VALPTR: REF DBG$VALDESC,! pointer to a value descriptor
;  637     0765  2         SIZE,                       ! Size of loop variable name
;  638     0766  2         SYMID_LIST,                 ! list of symids
;  639     0767  2         TEMP,                       ! temporary pointer to cis node
;  640     0768  2         TYPE,                       ! cis node type
;  641     0769  2         VALPTR: REF DBG$VALDESC,    ! pointer to a value descriptor
;  642     0770  2         VALUE,                      ! value in value descriptor
;  643     0771  2         VARNAME:REF VECTOR[,BYTE],! name for FOR loop var
;  644     0772  2         WHILE_FLAG;                 ! TRUE for WHILE cis
;  645     0773  2
;  646     0774  2
```

```
;  647          0775  2
;  648          0776  2        ! Decrement the count of the number of CIS levels we have.
;  649          0777  2        !
;  650          0778  2        DBG$GL_CIS_LEVELS = .DBG$GL_CIS_LEVELS - 1;
;  651          0779  2
;  652          0780  2
;  653          0781  2        ! If top link is an input buffer, release the storage for that buffer.
;  654          0782  2        !
;  655          0783  2        IF .DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_INPBUF
;  656          0784  2        THEN
;  657          0785  2            DBG$REL_MEMORY(.DBG$GL_CISHEAD[CIS$A_INIT_ADDR]);
;  658          0786  2
;  659          0787  2
;  660          0788  2        ! Also release storage for any other buffers that may have been
;  661          0789  2        ! allocated during processing of this line (new buffers get allocated
;  662          0790  2        ! when symbols defined by DEFINE/COMMAND are expanded).
;  663          0791  2        !
;  664          0792  2        BUFLIST = .DBG$GL_CISHEAD[CIS$A_BUFLIST];
;  665          0793  2        WHILE .BUFLIST NEQ 0 DO
;  666          0794  2            BEGIN
;  667          0795  2            DBG$REL_MEMORY(.BUFLIST[1]);
;  668          0796  2            TEMP = .BUFLIST[0];
;  669          0797  2            DBG$REL_MEMORY(.BUFLIST);
;  670          0798  2            BUFLIST = .TEMP;
;  671          0799  2            END;
;  672          0800  2        DBG$GL_CISHEAD[CIS$A_BUFLIST] = 0;
;  673          0801  2
;  674          0802  2
;  675          0803  2        ! If the top Command Input Steam Entry is a SCREEN CIS Entry, we must reset
;  676          0804  2        ! the screen displays to which print, source, and error output are directed
;  677          0805  2        ! to be the same as they were before this CIS Entry was added to the Command
;  678          0806  2        ! Input Stream.  We also reset the NOGO flag which disables STEP and GO
;  679          0807  2        ! commands inside screen display DEBUG command lists.
;  680          0808  2        !
;  681          0809  2        IF .DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL DBG$K_CIS_SCREEN
;  682          0810  2        THEN
;  683          0811  3            BEGIN
;  684          0812  3            DBG$GL_SCREEN_NOGO = .DBG$GL_CISHEAD[CIS$V_SCREEN_NOGO];
;  685          0813  3            DBG$GL_SCREEN_OUTPUT = .DBG$GL_CISHEAD[CIS$L_SCREEN_OUTPUT];
;  686          0814  3            DBG$GL_SCREEN_SOURCE = .DBG$GL_CISHEAD[CIS$L_SCREEN_SOURCE];
;  687          0815  3            DBG$GL_SCREEN_ERROR  = .DBG$GL_CISHEAD[CIS$L_SCREEN_ERROR];
;  688          0816  2            END;
;  689          0817  2
;  690          0818  2
;  691          0819  2        ! Unless we are exiting a loop or an indirect command procedure, handle
;  692          0820  2        ! the various looping constructs that have CIS entries.
;  693          0821  2        !
;  694          0822  2        IF NOT .EXIT_FLAG
;  695          0823  2        THEN
;  696          0824  2            BEGIN
;  697          0825  2
;  698          0826  2
;  699          0827  3            ! If the top link is a FOR CIS, then increment the FOR-loop counter.
;  700          0828  3            !
;  701          0829  3            IF .dbg$gl_cishead[cis$b_input_type] EQL cis_for
;  702          0830  3            THEN
;  703          0831  4                BEGIN
```

```
;  704          0832  4                bounds_match = FALSE;
;  705          0833  4
;  706          0834  4                ! Look up the loop counter.
;  707          0835  4                !
;  708          0836  4                varname = .dbg$gl_cishead [cis$a_for_loop_var];
;  709          0837  4                loop_incr = .dbg$gl_cishead [cis$l_for_loop_incr];
;  710          0838  4                IF dbg$def_sym_find (.varname, kind,
;  711          0839  4                                    valptr, global_flag, .message_vect)
;  712          0840  4                THEN
;  713          0841  4                    BEGIN
;  714          0842  5                    IF .kind EQL define_value
;  715          0843  5                    THEN
;  716          0844  5                        BEGIN
;  717          0845  6
;  718          0846  6                        value = .loop_incr + .valptr [dbg$l_value_value0];
;  719          0847  6                        IF (.loop_incr GTR 0
;  720          0848  7                           AND .value GTR .dbg$gl_cishead[cis$l_for_upper_bound])
;  721          0849  7                        OR (.loop_incr LSS 0
;  722          0850  7                           AND .value LSS .dbg$gl_cishead[cis$l_for_upper_bound])
;  723          0851  7                        THEN
;  724          0852  6                            bounds_match = TRUE
;  725          0853  6                        ELSE
;  726          0854  6                            BEGIN
;  727          0855  7
;  728          0856  7                            ! Copy the value descriptor. Fill in the new incremented
;  729          0857  7                            ! value into the copy. Save away the copy as the new
;  730          0858  7                            ! definition.
;  731          0859  7                            !
;  732          0860  7                            IF NOT dbg$nget_symid (.valptr, symid_list, .message_vect)
;  733          0861  7                            THEN
;  734          0862  7                                RETURN sts$k_severe;
;  735          0863  7                            IF NOT dbg$ncopy_desc (.valptr, new_valptr, .message_vect)
;  736          0864  7                            THEN
;  737          0865  7                                RETURN sts$k_severe;
;  738          0866  7                            dbg$sta_lock_symid (.symid_list);
;  739          0867  7                            new_valptr[dbg$l_value_value0] = .value;
;  740          0868  7                            ! Also copy the name.
;  741          0869  7                            new_name = dbg$get_memory (1+.varname[0]/4);
;  742          0870  7                            ch$move (1+.varname[0],.varname,.new_name);
;  743          0871  7                            IF NOT dbg$def_sym_add (.new_name, define_value,
;  744          0872  7                                                    .new_valptr, FALSE, dummy, .message_vect)
;  745          0873  7                            THEN
;  746          0874  7                                RETURN sts$k_severe;
;  747          0875  7                            dbg$gl_cishead [cis$w_length] =
;  748          0876  7                                .dbg$gl_cishead [cis$w_init_length];
;  749          0877  7                            dbg$gl_cishead [cis$a_input_ptr] =
;  750          0878  7                                .dbg$gl_cishead [cis$a_init_addr];
;  751          0879  7                            RETURN sts$k_success;
;  752          0880  7                            END;
;  753          0881  6                        END;
;  754          0882  5                    END;
;  755          0883  4                END;
;  756          0884  4
;  757          0885  4                ! Copy the loop variable name into temporary memory.
;  758          0886  4                ! This is for error-message purposes.
;  759          0887  4                !
;  760          0888  4                size = .varname[0];
```

```
;  761          0889  4              varname = dbg$get_tempmem (1+.size/4);
;  762          0890  4              ch$move (1+.size, .dbg$gl_cishead[cis$a_for_loop_var],
;  763          0891  4                               .varname);
;  764          0892  4
;  765          0893  4              ! If we fall through to here, we are exiting the loop for
;  766          0894  4              ! some reason.
;  767          0895  4              ! Release the space for the loop counter name.
;  768          0896  4              !
;  769          0897  4              dbg$rel_memory (.dbg$gl_cishead [cis$a_for_loop_var]);
;  770          0898  4
;  771          0899  4              ! If bounds_match is false, we are exiting the loop not because
;  772          0900  4              ! the lower bound has matched the upper bound, but rather because
;  773          0901  4              ! the loop variable had been redefined.
;  774          0902  4              !
;  775          0903  4              IF NOT .bounds_match
;  776          0904  4              THEN
;  777          0905  4                  SIGNAL (dbg$_loopvar, 1, .varname);
;  778          0906  3              END;
;  779          0907  3
;  780          0908  3          ! If the top link is a repeat cis, then decrement the count.
;  781          0909  3          !
;  782          0910  3          IF .dbg$gl_cishead[cis$b_input_type] EQL cis_repeat
;  783          0911  3          THEN
;  784          0912  4              BEGIN
;  785          0913  4              dbg$gl_cishead [cis$l_repeat_count] =
;  786          0914  4                  .dbg$gl_cishead [cis$l_repeat_count] - 1;
;  787          0915  4
;  788          0916  4              ! If the repeat count is greater than zero, reset the cis
;  789          0917  4              ! to the beginning of the action buffer.
;  790          0918  4              !
;  791          0919  4              IF .dbg$gl_cishead [cis$l_repeat_count] GTR 0
;  792          0920  4              THEN
;  793          0921  5                  BEGIN
;  794          0922  5                  dbg$gl_cishead [cis$w_length] =
;  795          0923  5                      .dbg$gl_cishead [cis$w_init_length];
;  796          0924  5                  dbg$gl_cishead [cis$a_input_ptr] =
;  797          0925  5                      .dbg$gl_cishead [cis$a_init_addr];
;  798          0926  5                  RETURN sts$k_success;
;  799          0927  4                  END;
;  800          0928  4
;  801          0929  3              END;
;  802          0930  3
;  803          0931  2          END;
;  804          0932  2
;  805          0933  2
;  806          0934  2          ! If the top link is a WHILE, or a REPEAT whose count has gone to zero,
;  807          0935  2          ! an IF CIS, a FOR CIS, or a SCREEN CIS, then release storage for the
;  808          0936  2          ! action buffer.  Here we subtract two from the address because storage
;  809          0937  2          ! was allocated as a counted string and included the count word.
;  810          0938  2          !
;  811          0939  2          IF .DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_WHILE  OR
;  812          0940  2             .DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_REPEAT OR
;  813          0941  2             .DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_IF     OR
;  814          0942  2             .DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_FOR    OR
;  815          0943  2             .DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_SCREEN
;  816          0944  2          THEN
;  817          0945  2              DBG$REL_MEMORY(.DBG$GL_CISHEAD[CIS$A_INIT_ADDR] - 2);
```

```
;  818     0946  2
;  819     0947
;  820     0948          ! If top link is a RAB, release the storage for the FAB, RAB and the
;  821     0949          ! buffer that holds the indirect command filespec.
;  822     0950          !
;  823     0951          IF .DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_RAB
;  824     0952          THEN
;  825     0953              BEGIN
;  826     0954
;  827     0955              LOCAL
;  828     0956                  FAB_PTR : REF $FAB_DECL, ! File access block pointer
;  829     0957                  RAB_PTR : REF $RAB_DECL; ! Record access block pointer
;  830     0958
;  831     0959              RAB_PTR = .DBG$GL_CISHEAD [ CIS$A_INPUT_PTR];
;  832     0960              FAB_PTR = .RAB_PTR [RAB$L_FAB];
;  833     0961              IF .DBG$GB_DEF_OUT [OUT_VERIFY]
;  834     0962              THEN
;  835     0963                  ICF_MESSAGE(2);     ! Exiting the ICF
;  836     0964
;  837     0965
;  838     0966              ! Release the filespec buffer. Remember this is a counted
;  839     0967              ! string so the address and length have to be adjusted to
;  840     0968              ! include the count.
;  841     0969              !
;  842     0970              DBG$REL_MEMORY (.FAB_PTR[FAB$L_FNA]-1);
;  843     0971
;  844     0972
;  845     0973              ! CLOSE and DISCONNECT
;  846     0974              !
;  847     0975              $CLOSE (FAB=.fab_ptr);
;  848     0976
;  849     0977              dbg$rel_memory (.rab_ptr);
;  850     0978              dbg$rel_memory (.fab_ptr);
;  851     0979
;  852     0980              ! Release the space taken up by the local define list.
;  853     0981              !
;  854     0982              IF NOT dbg$def_pr_exit (.message_vect)
;  855     0983              THEN
;  856     0984                  RETURN sts$k_severe;
;  857     0985
;  858     0986              END;
;  859     0987
;  860     0988          IF NOT .exit_flag
;  861     0989          THEN
;  862     0990              BEGIN
;  863     0991
;  864     0992
;  865     0993              ! For a WHILE CIS, find out whether the condition is still true.
;  866     0994              !
;  867     0995              IF .dbg$gl_cishead [cis$b_input_type] EQL cis_while
;  868     0996              THEN
;  869     0997  4               BEGIN
;  870     0998  4               while_flag = TRUE;
;  871     0999  4               cond = .dbg$gl_cishead [cis$v_while_flag];
;  872     1000  4               END
;  873     1001              ELSE
;  874     1002  3               while_flag = FALSE;
```

```
;  875          1003  3
;  876          1004  3          END;
;  877          1005  3
;  878          1006  3
;  879          1007  3      ! Remove the link from the command input stream
;  880          1008  3      !
;  881          1009  3      temp = .dbg$gl_cishead ;
;  882          1010  3      dbg$gl_cishead = .dbg$gl_cishead [cis$a_next_link];
;  883          1011  3
;  884          1012  3      ! Now release the storage for the link itself
;  885          1013  3      !
;  886          1014  3      dbg$rel_memory (.temp);
;  887          1015  3
;  888          1016  3      IF NOT .exit_flag
;  889          1017  3      THEN
;  890          1018  3          ! If the cis is a WHILE, then set up the top cis for another iteration.
;  891          1019  3          !
;  892          1020  3          IF .while_flag
;  893          1021  3          THEN
;  894          1022  3              IF .cond
;  895          1023  3              THEN
;  896          1024  3                  BEGIN
;  897          1025  3                  dbg$gl_cishead [cis$a_input_ptr] =
;  898          1026  3                      .dbg$gl_cishead [cis$a_while_clause];
;  899          1027  3                  dbg$gl_cishead [cis$w_length] =
;  900          1028  3                      .dbg$gl_cishead [cis$w_while_length];
;  901          1029  3                  END;
;  902          1030  3
;  903          1031  2      RETURN sts$k_success;
;  904          1032  2
;  905          1033  1      END;


                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                                 08  00023 P.AAD:  .BYTE   8                                                    ;
                                  67  6E  69  72  65  74  6E  65  00024          .ASCII  \entering\                                 ;
                                                 07  0002C P.AAE:  .BYTE   7                                                    ;
                                      67  6E  69  74  69  78  65  0002D          .ASCII  \exiting\                                  ;

                                                                    ENTER_PHRASE=       P.AAD
                                                                    EXIT_PHRASE=        P.AAE
                                                        .EXTRN  SYS$CLOSE

                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                     0FFC 00000         .ENTRY  DBG$NCIS_REMOVE, Save R2,R3,R4,R5,R6,R7,R8,-; 0718
                                                                                R9,R10,R11                                          ;
                                      5B 00000000G   00  9E 00002         MOVAB   DBG$GL_CISHEAD, R11                            ;
                                      5E             18  C2 00009         SUBL2   #24, SP                                        ;
                                         00000000G   00  D7 0000C         DECL    DBG$GL_CIS_LEVELS                              ; 0778
                                      50             6B  D0 00012         MOVL    DBG$GL_CISHEAD, R0                             ; 0783
                                      02       02    A0  91 00015         CMPB    2(R0), #2                                      ;
                                                     0A  12 00019         BNEQ    1$                                             ;
                                               0C    A0  DD 0001B         PUSHL   12(R0)                                         ; 0785
                                  00000000G    00        01  FB 0001E         CALLS   #1, DBG$REL_MEMORY                         ;
```

```
                                 50        6B  D0 00025 1$:     MOVL    DBG$GL_CISHEAD, R0                                      ; 0792
                                 52    30  A0  D0 00028         MOVL    48(R0), BUFLIST                                         ;
                                       1B  13 0002C 2$:        BEQL    3$                                                      ; 0793
                                   04  A2  DD 0002E           PUSHL   4(BUFLIST)                                              ; 0795
                    00000000G    00       01  FB 00031         CALLS   #1, DBG$REL_MEMORY
                                 5A       62  D0 00038         MOVL    (BUFLIST), TEMP                                         ; 0796
                                          52  DD 0003B         PUSHL   BUFLIST                                                 ; 0797
                    00000000G    00       01  FB 0003D         CALLS   #1, DBG$REL_MEMORY
                                 52       5A  D0 00044         MOVL    TEMP, BUFLIST                                           ; 0798
                                          E3  11 00047         BRB     2$                                                      ; 0793
                                 50        6B  D0 00049 3$:     MOVL    DBG$GL_CISHEAD, R0                                      ; 0800
                                   30      A0  D4 0004C         CLRL    48(R0)
                                 08    02  A0  91 0004F         CMPB    2(R0), #8                                               ; 0809
                                          22  12 00053         BNEQ    4$
00000000G  00        12    A0             01  02  EF 00055         EXTZV   #2, #1, 18(R0), DBG$GL_SCREEN_NOGO                      ; 0812
                    00000000G    00   24  A0  D0 0005F         MOVL    36(R0), DBG$GL_SCREEN_OUTPUT                            ; 0813
                    00000000G    00   28  A0  D0 00067         MOVL    40(R0), DBG$GL_SCREEN_SOURCE                            ; 0814
                    00000000G    00   2C  A0  D0 0006F         MOVL    44(R0), DBG$GL_SCREEN_ERROR                             ; 0815
                                 03    04  AC  E9 00077 4$:     BLBC    EXIT_FLAG, 5$                                           ; 0822
                                       0121 31 0007B         BRW     16$
                                 07    02  A0  91 0007E 5$:     CMPB    2(R0), #7                                               ; 0829
                                          03  13 00082         BEQL    6$
                                       00FE 31 00084         BRW     14$
                                          59  D4 00087 6$:     CLRL    BOUNDS_MATCH                                            ; 0833
                                 57    1C  A0  D0 00089         MOVL    28(R0), VARNAME                                         ; 0837
                                 53    20  A0  D0 0008D         MOVL    32(R0), LOOP_INCR                                       ; 0838
                                       08  AC  DD 00091         PUSHL   MESSAGE_VECT                                            ; 0840
                                       04  AE  9F 00094         PUSHAB  GLOBAL_FLAG                                             ; 0839
                                       0C  AE  9F 00097         PUSHAB  VALPTR
                                       14  AE  9F 0009A         PUSHAB  KIND
                                          57  DD 0009D         PUSHL   VARNAME
                    00000000G    00       05  FB 0009F         CALLS   #5, DBG$DEF_SYM_FIND
                                 2C       50  E9 000A6         BLBC    R0, 9$
                                 05    08  AE  D1 000A9         CMPL    KIND, #5                                                ; 0843
                                          26  12 000AD         BNEQ    9$
                                 52    04  AE  D0 000AF         MOVL    VALPTR, R2                                              ; 0847
                54               53    20  A2  C1 000B3         ADDL3   32(R2), LOOP_INCR, VALUE
                                          53  D5 000B8         TSTL    LOOP_INCR                                               ; 0848
                                          09  15 000BA         BLEQ    7$
                                 50        6B  D0 000BC         MOVL    DBG$GL_CISHEAD, R0                                      ; 0849
                              18  A0          54  D1 000BF         CMPL    VALUE, 24(R0)
                                          0D  14 000C3         BGTR    8$
                                          53  D5 000C5 7$:     TSTL    LOOP_INCR                                               ; 0850
                                          0E  18 000C7         BGEQ    10$
                                 50        6B  D0 000C9         MOVL    DBG$GL_CISHEAD, R0                                      ; 0851
                              18  A0          54  D1 000CC         CMPL    VALUE, 24(R0)
                                          05  18 000D0         BGEQ    10$
                                 59        01  D0 000D2 8$:     MOVL    #1, BOUNDS_MATCH                                        ; 0853
                                          72  11 000D5 9$:     BRB     13$
                                       08  AC  DD 000D7 10$:    PUSHL   MESSAGE_VECT                                            ; 0861
                                       10  AE  9F 000DA         PUSHAB  SYMID_LIST
                                          52  DD 000DD         PUSHL   R2
                    00000000G    00       03  FB 000DF         CALLS   #3, DBG$NGET_SYMID
                                 55       50  E9 000E6         BLBC    R0, 11$
                                       08  AC  DD 000E9         PUSHL   MESSAGE_VECT                                            ; 0864
                                       14  AE  9F 000EC         PUSHAB  NEW_VALPTR
                                          52  DD 000EF         PUSHL   R2
```

```
                00000000G  00              03  FB  000F1          CALLS   #3, DBG$NCOPY_DESC                      ;
                           43              50  E9  000F8          BLBC    R0, 11$                                 ;
                                        0C AE  DD  000FB          PUSHL   SYMID_LIST                              ; 0867
                00000000G  00              01  FB  000FE          CALLS   #1, DBG$STA_LOCK_SYMID                  ;
                           56           10 AE  D0  00105          MOVL    NEW_VALPTR, R6                          ; 0868
                      20   A6              54  D0  00109          MOVL    VALUE, 32(R6)                           ;
                           50              67  9A  0010D          MOVZBL  (VARNAME), R0                           ; 0870
                           50              04  C6  00110          DIVL2   #4, R0                                  ;
                                        01 A0  9F  00113          PUSHAB  1(R0)                                   ;
                00000000G  00              01  FB  00116          CALLS   #1, DBG$GET_MEMORY                      ;
                           58              50  D0  0011D          MOVL    R0, NEW_NAME                            ;
                           50              67  9A  00120          MOVZBL  (VARNAME), R0                           ; 0871
                                           50  D6  00123          INCL    R0                                      ;
                68         67              50  28  00125          MOVC3   R0, (VARNAME), (NEW_NAME)               ;
                                        08 AC  DD  00129          PUSHL   MESSAGE_VECT                            ; 0873
                                        18 AE  9F  0012C          PUSHAB  DUMMY                                   ; 0872
                                           7E  D4  0012F          CLRL    -(SP)                                   ;
                                           56  DD  00131          PUSHL   R6                                      ; 0873
                                           05  DD  00133          PUSHL   #5                                      ; 0872
                                           58  DD  00135          PUSHL   NEW_NAME                                ;
                00000000G  00              06  FB  00137          CALLS   #6, DBG$DEF_SYM_ADD                     ;
                           03              50  E8  0013E 11$:      BLBS    R0, 12$                                 ;
                                        00F2 31  00141          BRW     20$                                     ;
                           50              6B  D0  00144 12$:      MOVL    DBG$GL_CISHEAD, R0                      ; 0876
                                           4A  11  00147          BRB     15$                                     ; 0877
                           52              67  9A  00149 13$:      MOVZBL  (VARNAME), SIZE                         ; 0888
                50         52              04  C7  0014C          DIVL3   #4, SIZE, R0                            ; 0889
                                        01 A0  9F  00150          PUSHAB  1(R0)                                   ;
                00000000G  00              01  FB  00153          CALLS   #1, DBG$GET_TEMPMEM                     ;
                           57              50  D0  0015A          MOVL    R0, VARNAME                             ;
                                           52  D6  0015D          INCL    R2                                      ; 0890
                           56              6B  D0  0015F          MOVL    DBG$GL_CISHEAD, R6                      ;
                67      1C B6              52  28  00162          MOVC3   R2, @28(R6), (VARNAME)                  ; 0891
                                        1C A6  DD  00167          PUSHL   28(R6)                                  ; 0897
                00000000G  00              01  FB  0016A          CALLS   #1, DBG$REL_MEMORY                      ;
                           11              59  E8  00171          BLBS    BOUNDS_MATCH, 14$                       ; 0903
                                           57  DD  00174          PUSHL   VARNAME                                 ; 0905
                                           01  DD  00176          PUSHL   #1                                      ;
                                000286C3   8F  DD  00178          PUSHL   #165571                                 ;
                00000000G  00              03  FB  0017E          CALLS   #3, LIB$SIGNAL                          ;
                           50              6B  D0  00185 14$:      MOVL    DBG$GL_CISHEAD, R0                      ; 0910
                           04           02 A0  91  00188          CMPB    2(R0), #4                               ;
                                           11  12  0018C          BNEQ    16$                                     ;
                                        18 A0  D7  0018E          DECL    24(R0)                                  ; 0914
                                           0C  15  00191          BLEQ    16$                                     ; 0919
                           60           10 A0  B0  00193 15$:      MOVW    16(R0), (R0)                            ; 0923
                      04   A0           0C A0  D0  00197          MOVL    12(R0), 4(R0)                           ; 0925
                                        00DE 31  0019C          BRW     24$                                     ; 0926
                           50              6B  D0  0019F 16$:      MOVL    DBG$GL_CISHEAD, R0                      ; 0939
                           51           02 A0  9A  001A2          MOVZBL  2(R0), R1                               ;
                           05              51  91  001A6          CMPB    R1, #5                                  ;
                                           14  13  001A9          BEQL    17$                                     ;
                           04              51  91  001AB          CMPB    R1, #4                                  ; 0940
                                           0F  13  001AE          BEQL    17$                                     ;
                           06              51  91  001B0          CMPB    R1, #6                                  ; 0941
                                           0A  13  001B3          BEQL    17$                                     ;
                           07              51  91  001B5          CMPB    R1, #7                                  ; 0942
```

```
                                        05  13 001B8          BEQL    17$                                     ;
                        08              51  91 001BA          CMPB    R1, #8                                  ; 0943
                                        0C  12 001BD          BNEQ    18$                                     ;
                7E      0C      A0      02  C3 001BF 17$:     SUBL3   #2, 12(R0), -(SP)                       ; 0945
                00000000G       00      01  FB 001C4          CALLS   #1, DBG$REL_MEMORY                      ;
                        50              6B  D0 001CB 18$:     MOVL    DBG$GL_CISHEAD, R0                      ; 0951
                        01      02      A0  91 001CE          CMPB    2(R0), #1                               ;
                                        66  12 001D2          BNEQ    21$                                     ;
                        53      04      A0  D0 001D4          MOVL    4(R0), RAB_PTR                          ; 0959
                        52      3C      A3  D0 001D8          MOVL    60(RAB_PTR), FAB_PTR                    ; 0960
                        1F  00000000G   00  E9 001DC          BLBC    DBG$GB_DEF_OUT+2, 19$                   ; 0961
                        50  00000000'   EF  9E 001E3          MOVAB   EXIT_PHRASE, PHRASE                     ; 0963
                                2C      A2  DD 001EA          PUSHL   44(FAB_PTR)                             ;
                        7E      34      A2  9A 001ED          MOVZBL  52(FAB_PTR), -(SP)                      ;
                                        50  DD 001F1          PUSHL   PHRASE                                  ;
                                        03  DD 001F3          PUSHL   #3                                      ;
                            0002808B    8F  DD 001F5          PUSHL   #163979                                 ;
                00000000G       00      05  FB 001FB          CALLS   #5, DBG$NOUT_INFO                       ;
                7E      2C      A2      01  C3 00202 19$:     SUBL3   #1, 44(FAB_PTR), -(SP)                  ; 0970
                00000000G       00      01  FB 00207          CALLS   #1, DBG$REL_MEMORY                      ;
                                        52  DD 0020E          PUSHL   FAB_PTR                                 ; 0975
                00000000G       00      01  FB 00210          CALLS   #1, SYS$CLOSE                           ;
                                        53  DD 00217          PUSHL   RAB_PTR                                 ; 0977
                00000000G       00      01  FB 00219          CALLS   #1, DBG$REL_MEMORY                      ;
                                        52  DD 00220          PUSHL   FAB_PTR                                 ; 0978
                00000000G       00      01  FB 00222          CALLS   #1, DBG$REL_MEMORY                      ;
                        08      AC      DD 00229          PUSHL   MESSAGE_VECT                            ; 0982
                00000000G       00      01  FB 0022C          CALLS   #1, DBG$DEF_PR_EXIT                     ;
                        04              50  E8 00233          BLBS    R0, 21$                                 ;
                        50              04  D0 00236 20$:     MOVL    #4, R0                                  ; 0984
                                        04  00239            RET                                             ;
                        16      04      AC  E8 0023A 21$:     BLBS    EXIT_FLAG, 23$                          ; 0988
                        50              6B  D0 0023E          MOVL    DBG$GL_CISHEAD, R0                      ; 0995
                        05      02      A0  91 00241          CMPB    2(R0), #5                               ;
                                        0B  12 00245          BNEQ    22$                                     ;
                        52              01  D0 00247          MOVL    #1, WHILE_FLAG                          ; 0998
        53      12      A0              01  01  EF 0024A      EXTZV   #1, #1, 18(R0), COND                    ; 0999
                                        02  11 00250          BRB     23$                                     ; 0995
                                        52  D4 00252 22$:     CLRL    WHILE_FLAG                              ; 1002
                        50              6B  D0 00254 23$:     MOVL    DBG$GL_CISHEAD, R0                      ; 1009
                        5A              50  D0 00257          MOVL    R0, TEMP                                ;
                        6B      08      A0  D0 0025A          MOVL    8(R0), DBG$GL_CISHEAD                   ; 1010
                                        5A  DD 0025E          PUSHL   TEMP                                    ; 1014
                00000000G       00      01  FB 00260          CALLS   #1, DBG$REL_MEMORY                      ;
                        12      04      AC  E8 00267          BLBS    EXIT_FLAG, 24$                          ; 1016
                        0F              52  E9 0026B          BLBC    WHILE_FLAG, 24$                         ; 1020
                        0C              53  E9 0026E          BLBC    COND, 24$                               ; 1022
                        50              6B  D0 00271          MOVL    DBG$GL_CISHEAD, R0                      ; 1025
                    04  A0      14      A0  D0 00274          MOVL    20(R0), 4(R0)                           ; 1026
                        60      34      A0  B0 00279          MOVW    52(R0), (R0)                            ; 1028
                        50              01  D0 0027D 24$:     MOVL    #1, R0                                  ; 1031
                                        04  00280            RET                                             ; 1033

; Routine Size:  641 bytes,    Routine Base:  DBG$CODE + 031E
```

```
;  907          1034  1 GLOBAL ROUTINE DBG$NGET_ADDRESS (ADDR_EXP_DESC, ADDRESS, TYPE, PROLOG_FLAG, MESSAGE_VECT) =
;  908          1035  1
;  909          1036  1 !++
;  910          1037  1 ! FUNCTIONAL DESCRIPTION:
;  911          1038  1 !
;  912          1039  1 !       This routine is called with a descriptor, as returned
;  913          1040  1 !       by the Address Expression Interpreter, to obtain the address bound to the
;  914          1041  1 !       entity described by the descriptor.
;  915          1042  1 !
;  916          1043  1 ! FORMAL PARAMETERS:
;  917          1044  1 !
;  918          1045  1 !       ADDR_EXP_DESC   - A longword containing the address of either a
;  919          1046  1 !                         value or primary descriptor
;  920          1047  1 !
;  921          1048  1 !       ADDRESS         - The address of a quadword to contain the resulting
;  922          1049  1 !                         byte address and bit offset
;  923          1050  1 !
;  924          1051  1 !       TYPE            - The address of a longword to contain the type of the address
;  925          1052  1 !                         (No longer used).
;  926          1053  1 !
;  927          1054  1 !       PROLOG_FLAG     - A flag set to true to indicate this routine is
;  928          1055  1 !                         called from SET BREAK/TRACE, SHOW BREAK/TRACE, where
;  929          1056  1 !                         routine break address is taken from the primary
;  930          1057  1 !                         routine/entry rst entry.
;  931          1058  1 !
;  932          1059  1 !       MESSAGE_VECT    - The address of a longword to contain the address of a
;  933          1060  1 !                         message argument vector upon detection of errors
;  934          1061  1 !
;  935          1062  1 ! IMPLICIT INPUTS:
;  936          1063  1 !
;  937          1064  1 !       NONE
;  938          1065  1 !
;  939          1066  1 ! IMPLICIT OUTPUTS:
;  940          1067  1 !
;  941          1068  1 !       On error, a message argument vector is constructed and returned.
;  942          1069  1 !
;  943          1070  1 ! ROUTINE VALUE:
;  944          1071  1 !
;  945          1072  1 !       An unsigned integer longword completion code
;  946          1073  1 !
;  947          1074  1 ! COMPLETION CODES:
;  948          1075  1 !
;  949          1076  1 !       STS$K_SUCCESS (1)       - Success. Address and type returned.
;  950          1077  1 !
;  951          1078  1 !       STS$K_SEVERE  (4)       - Failure. No type and/or address obtained.
;  952          1079  1 !                                 Message argument vector returned.
;  953          1080  1 !
;  954          1081  1 ! SIDE EFFECTS:
;  955          1082  1 !
;  956          1083  1 !       NONE
;  957          1084  1 !
;  958          1085  1 !--
;  959          1086  2     BEGIN
;  960          1087  2     MAP
;  961          1088  2         ADDRESS: REF VECTOR[,LONG],
;  962          1089  2         ADDR_EXP_DESC: REF DBG$VALDESC; ! Points to a new style Descriptor.
;  963          1090  2
```

```
;  964          1091  2     LOCAL
;  965          1092  2         VMS_DESC: REF DBG$STG_DESC,
;  966          1093  2         RSTPTR: REF RST$ENTRY;
;  967          1094  2
;  968          1095  2
;  969          1096  2     ! If the flag is set, take the break address from Routine/Entry RST
;  970          1097  2     ! in Primary.  (The only way this flag can be set is in DBGEVENT.)
;  971          1098  2     !
;  972          1099  2     IF .PROLOG_FLAG
;  973          1100  2     THEN
;  974          1101  2         BEGIN
;  975          1102  2         RSTPTR = .ADDR_EXP_DESC[DBG$L_DHDR_SYMIDO];
;  976          1103  2         ADDRESS[0] = .RSTPTR[RST$L_BREAKADDR];
;  977          1104  2         ADDRESS[1] = 0;
;  978          1105  2         RETURN sts$k_success;
;  979          1106  2         END;
;  980          1107  2
;  981          1108  2
;  982          1109  2     ! Check whether we are looking at a Primary Descriptor.
;  983          1110  2     !
;  984          1111  2     IF  .ADDR_EXP_DESC [DBG$B_DHDR_TYPE] EQL DBG$K_PRIMARY_DESC
;  985          1112  2     THEN
;  986          1113  2         BEGIN
;  987          1114  2
;  988          1115  2
;  989          1116  2         ! Allocate temporary memory for the VMS descriptor.
;  990          1117  2         !
;  991          1118  2         VMS_DESC = DBG$GET_TEMPMEM (3);
;  992          1119  2
;  993          1120  2         ! Call the routine that fills in the VMS descriptor.
;  994          1121  2         !
;  995          1122  2         DBG$MAKE_VMS_DESC (.ADDR_EXP_DESC, .VMS_DESC);
;  996          1123  2         END
;  997          1124  2
;  998          1125  2     ! Check for Volatile Value Descriptor.
;  999          1126  2     !
; 1000          1127  2     ELSE
; 1001          1128  2         IF .ADDR_EXP_DESC [DBG$B_DHDR_TYPE] EQL DBG$K_V_VALUE_DESC
; 1002          1129  2         THEN
; 1003          1130  2             VMS_DESC = ADDR_EXP_DESC [DBG$A_VALUE_VMSDESC]
; 1004          1131  2
; 1005          1132  2         ! Any other kind of descriptor is an error.
; 1006          1133  2         !
; 1007          1134  2         ELSE
; 1008          1135  2             $DBG_ERROR ('DBGNEXCTE\DBG$NGET_ADDRESS unexpected descriptor type');
; 1009          1136  2
; 1010          1137  2     ! Fill in the output parameter to point to the
; 1011          1138  2     ! (byte address, bit offset) quadword in the VMS descriptor.
; 1012          1139  2     !
; 1013          1140  2     ADDRESS[0] = .VMS_DESC[DSC$A_POINTER];
; 1014          1141  2     IF .VMS_DESC[DSC$B_CLASS] NEQ DSC$K_CLASS_UBS
; 1015          1142  2     THEN
; 1016          1143  2         ADDRESS[1] = 0
; 1017          1144  2     ELSE
; 1018          1145  2         ADDRESS[1] = .VMS_DESC[DSC$L_POS];
; 1019          1146  2
; 1020          1147  2     RETURN sts$k_success;
```

```
; 1021          1148  1     END;                    ! End of dbg$nget_address


                                                    .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

24 47 42 44 5C 45 54 43 58 45 4E 47 42 44 35  00034 P.AAF:  .ASCII  \5DBGNEXCTE\<92>\DBG$NGET_ADDRESS unexpe\  ;
6E 75 20 53 53 45 52 44 44 41 5F 54 45 47 4E  00043                                                          ;
                                 65 70 78 65  00052                                                          ;
72 6F 74 70 69 72 63 73 65 64 20 64 65 74 63  00056          .ASCII  \cted descriptor type\                 ;
                                 65 70 79 74 20  00065                                                      ;


                                                    .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                      0004 00000            .ENTRY  DBG$NGET_ADDRESS, Save R2                ; 1034
                              12      10   AC   E9 00002    BLBC    PROLOG_FLAG, 1$                          ; 1099
                              50      04   AC   D0 00006    MOVL    ADDR_EXP_DESC, R0                        ; 1102
                              51      0C   A0   D0 0000A    MOVL    12(R0), RSTPTR                           ;
                              50      08   AC   D0 0000E    MOVL    ADDRESS, R0                              ; 1103
                              60      28   A1   D0 00012    MOVL    40(RSTPTR), (R0)                         ;
                                           5D   11 00016    BRB     5$                                       ; 1104
00000079   8F       04   BC          08        10   ED 00018 1$:     CMPZV   #16, #8, @ADDR_EXP_DESC, #121    ; 1111
                                           1A   12 00022    BNEQ    2$                                       ;
                                           03   DD 00024    PUSHL   #3                                       ; 1118
                    00000000G  00          01   FB 00026    CALLS   #1, DBG$GET_TEMPMEM                      ;
                              52           50   D0 0002D    MOVL    R0, VMS_DESC                             ;
                                           52   DD 00030    PUSHL   VMS_DESC                                 ; 1122
                                      04   AC   DD 00032    PUSHL   ADDR_EXP_DESC                            ;
                    00000000G  00          02   FB 00035    CALLS   #2, DBG$MAKE_VMS_DESC                    ;
                                           28   11 0003C    BRB     4$                                       ; 1111
00000083   8F       04   BC          08        10   ED 0003E 2$:     CMPZV   #16, #8, @ADDR_EXP_DESC, #131    ; 1128
                                           07   12 00048    BNEQ    3$                                       ;
                    52       04   AC                      14   C1 0004A    ADDL3   #20, ADDR_EXP_DESC, VMS_DESC ; 1130
                                           15   11 0004F    BRB     4$                                       ;
                                 00000000'    EF   9F 00051 3$:     PUSHAB  P.AAF                            ; 1135
                                           01   DD 00057    PUSHL   #1                                       ;
                                 00028362     8F   DD 00059    PUSHL   #164706                                  ;
                    00000000G  00          03   FB 0005F    CALLS   #3, LIB$SIGNAL                           ;
                          08   BC      04   A2   D0 00066 4$:     MOVL    4(VMS_DESC), @ADDRESS                ; 1140
                              50      08   AC   D0 0006B    MOVL    ADDRESS, R0                              ; 1143
                              0D      03   A2   91 0006F    CMPB    3(VMS_DESC), #13                         ; 1141
                                           05   13 00073    BEQL    6$                                       ;
                                      04   A0   D4 00075 5$:     CLRL    4(R0)                                ; 1143
                                           05   11 00078    BRB     7$                                       ;
                          04   A0      08   A2   D0 0007A 6$:     MOVL    8(VMS_DESC), 4(R0)                   ; 1145
                              50           01   D0 0007F 7$:     MOVL    #1, R0                               ; 1147
                                           04   00082            RET                                      ; 1148

; Routine Size:  131 bytes,    Routine Base:  DBG$CODE + 059F

; 1022          1149  1 END                                   !End of module
; 1023          1150  0 ELUDOM
```

```
                                                .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  DBG$PLIT                       106  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$CODE                      1570  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)


;                           Library Statistics
;
;                                    --------  Symbols  --------      Pages       Processing
;       File                         Total     Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1      18619         23         0        1000       00:01.9
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1
;                                        32          0         0           7       00:00.1
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1
;                                      1545        167        10          97       00:01.9
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                       418          0         0          31       00:00.4
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1
;                                       386          4         1          22       00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGGEN.L32;1
;                                       150          2         1          12       00:00.3



;                               COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:DBGNEXCTE/OBJ=OBJ$:DBGNEXCTE MSRC$:DBGNEXCTE/UPDATE=(ENH$:DBGNEXCTE)

; Size:         1570 code + 106 data bytes
; Run Time:       00:32.2
; Elapsed Time:   01:41.7
; Lines/CPU Min:   2146
; Lexemes/CPU-Min: 12100
; Memory Used:  261 pages
; Compilation Complete
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DBGNERMSG
LIS

DBGNHELP
LIS

DBGNPARSE
LIS

DBGNEXCTE
LIS

DBGNPNP
LIS